НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

(повне найменування інституту, факультету)

Автоматизованих систем обробки інформації і управління

(повна назва кафедри)

«До захисту допущено»
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (ініціали, прізвище)
“ ” 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Програмне забезпечення інформаційних управляючих систем та технологій»

спеціальності «121 Інженерія програмного забезпечення»

на тему *Веб-застосування моніторингу життєвих подій*

Виконав: студент IV курсу, групи *ІП-63 Семченко Андрій Олегович* _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *ст. викладач Ковтунець Олесь Володимирович* _____
посада,науковий ступінь,вчене звання,прізвище,і ім’я, по батькові (підпис)

Консультант з графічної документації *доц., к.т.н., Ліщук К.І.* _____
посада,науковий ступінь,вчене звання,прізвище,і ім’я, по батькові (підпис)

Рецензент:

доц., к.т.н., доц. Пасько В.П.
посада,науковий ступінь,вчене звання,прізвище,ім’я, по батькові (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – *121 Інженерія програмного забезпечення*

Освітньо-професійна програма – *Програмне забезпечення інформаційних управляючих систем та технологій*

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис)
“___”_____2020 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ

Семченку Андрію Олеговичу
(прізвище, ім’я, по батькові)

1. Тема проєкту «*Веб-застосування моніторингу життєвих подій*»

керівник проєкту Ковтунець Олесь Володимирович ст.викл.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “07” травня 2020 р. №1081-с

2. Термін подання студентом проєкту «08» *червня 2020 року*

3. Вихідні дані до проєкту
Технічне завдання

4. Зміст пояснювальної записки
1) Аналіз вимог до програмного забезпечення: основні визначення та терміни, опис предметного середовища, огляд існуючих технічних рішень та відомих програмних продуктів, розробка функціональних та нефункціональних вимог
2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, засоби розробки, технічні рішення, архітектура програмного забезпечення
3) Розгортання та впровадження програмного забезпечення
4) Керівництво користувача, методика випробувань програмного продукту

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема бази даних

3) Креслення вигляду екранних форм

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «10» *березня 2020 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	*11.03.2020*	
2.	*Аналіз існуючих методів розв'язання задачі*	*18.03.2020*	
3.	*Постановка та формалізація задачі*	*25.03.2020*	
4.	*Аналіз вимог до програмного забезпечення*	*01.04.2020*	
5.	*Алгоритмізація задачі*	*08.04.2020*	
6.	*Моделювання програмного забезпечення*	*15.04.2020*	
7.	*Обґрунтування використовуваних технічних засобів*	*22.04.2020*	
8.	*Розробка архітектури програмного забезпечення*	*29.04.2020*	
9.	*Розробка програмного забезпечення*	*06.05.2020*	
10.	*Налагодження програми*	*13.05.2020*	
11.	*Виконання графічних документів*	*20.05.2020*	
12.	*Оформлення пояснювальної записки*	*27.05.2020*	
13.	*Подання ДП на попередній захист*	*28.05.2020*	
14.	*Подання ДП рецензенту*		
15.	*Подання ДП на основний захист*	*08.06.2020*	

Студент _____ Андрій СЕМЧЕНКО
(підпис)

Керівник _____ Олесь КОВТУНЕЦЬ
(підпис)

№з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-6326.045420.02.81	Пояснювальна записка	73	
3	А4	КПІ.ІП-6326.045420.03.91	Технічне завдання	11	
4	А4	КПІ.ІП-6326.045420.04.13	Опис програми	51	
5	А4	КПІ.ІП-6326.045420.05.51	Програма та методика тестування	6	
6	А4	КПІ.ІП-6326.045420.06.34	Керівництво користувача	19	
			Матеріали графічні		
7	А3	КПІ.ІП-6326.045420.07.99.СС	Схема варіантів використання системи	1	
8	А3	КПІ.ІП-6326.045420.08.99.СБД	Схема бази даних	1	
9	А3	КПІ.ІП-6326.045420.09.99.КЕ	Креслення вигляду екранних форм	1	
10	А3	КПІ.ІП-6326.045420.10.99.СС	Схема структурна класів програмного забезпечення	1	

Зм.	Арк.	ПІБ	Підп.	Дата	КПІ.ІП-6326.045420.00.90		
Розробн.		Семченко А.О.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Ковтунець О.В.					
Консульт						КПІ ім.Ігоря Сікорського кафедра АСОІУ гр. ІП-63	
Н/контр.		Ліщук К.І.					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка
до дипломного проєкту

на тему: Веб-застосування моніторингу життєвих подій

Київ – 2020 року

АНОТАЦІЯ

Пояснювальна записка до дипломного проекту: 11 рис., 45 табл., 14 джерел – загалом 73 сторінки.

Об’єкт дослідження: інструменти зберігання та аналізу життєвих подій.

Мета дипломного проекту: надати користувачу можливість зберігати інформацію про життєві події у структурованому вигляді та аналізувати їх

У першому розділі була проаналізована предметна область та вимоги до програмного забезпечення. Були розроблені функціональні вимоги, опис варіантів використання та їх структурна схема. Також у розділі були описані нефункціональні вимоги до системи.

У другому розділі описана архітектура, діаграма класів, специфікація методів, специфікація полів, використані підходи та інструменти до написання програми. А також розроблена структура бази даних.

У третьому розділі наведений розроблений план та проведення тестування, а також описані результати самого тестування.

У четвертому розділі представлено, яким чином розгортати дане програмне забезпечення. Також наведена інструкція користувача.

КЛЮЧОВІ СЛОВА: СТРУКТУРУВАННЯ ДАНИХ, АНАЛІЗ ДАНИХ, МОНІТОРИНГ, БАЗА ЗНАНЬ, СЕРВЕРНІ СИСТЕМИ, ВЕБ ЗАСТОСУНОК.

ABSTRACT

Explanatory note to the degree project: 11 fig., 45 tables, 14 references – totally 73 pages.

The object of study: tools for life events storing and analyzing.

The aim of the degree project: provide to user an ability to store information about life events in structured way and analyze it.

In the first section, the domain and software requirements were analyzed. Functional requirements model, use cases description and use-case diagram were developed. Also, non-functional requirements were described.

In the second section architecture diagram, class diagrams, method specifications, field specifications, approaches, and software development tools were described. Database ER diagram was also developed.

In the third section plan of testing and describes the results of the test.

The fourth section presents how to deploy this software. There is a user manual.

KEYWORDS: DATA STRUCTURING, DATA ANYLYZING, SERVER SYSTEMS, WEB APPLICATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення.

KPI (Key Performance Indicator) – фінансова та нефінансова система оцінки, яка допомагає організації визначити досягнення стратегічних цілей.

PDCA (Plan-Do-Act-Check) – модель безперервного поліпшення процесів, цикл PDCA — плануй (Plan), роби (Do), перевіряй (Check), впливай (Act). При її застосуванні в різноманітних областях діяльності (наприклад, управління якістю) дозволяє ефективно керувати цією діяльністю на системній основі.

UI (User Interface) — засіб зручної взаємодії користувача з інформаційною системою.

UX (User Experience) — це те, що людина відчуває при користуванні продуктом, системою чи сервісом (послугою). Основними об'єктами дослідження є враження, емоції та користь, отримані від взаємодії з продуктом.

БЗ (База знань) — це особливого роду база даних, розроблена для управління знаннями (метаданими), тобто збором, зберіганням, пошуком і видачею знань.

Онтологія — формалізоване представлення знань про певну предметну область (середовище, світ), придатне для автоматизованої обробки.

Інженерія знань — область штучного інтелекту, пов'язана із розробкою експертних систем і баз знань. Вивчає методи і засоби для отримання, представлення, структурування і використання знань.

СКУД (Система Керування Базами Даних) — набір взаємопов'язаних даних (база даних) і програм для доступу до цих даних.

Diary Software – клас програмного забезпечення, що призначене для фіксація побаченої, почутої, внутрішньої пережитої події, яка щойно сталася. Виконує ті ж функції, що і паперовий щоденник.

BPMN (Business Process Model and Notation) — система умовних позначень (нотація) для моделювання бізнес-процесів.

ER-модель (Entity-relationship модель) — модель даних, яка дозволяє описувати концептуальні схеми за допомогою узагальнених конструкцій блоків.

DTO (Data Transfer Object) — один із шаблонів проектування, який використовують для передачі даних між підсистемами програми.

Angular — написаний на мові програмування TypeScript фреймворк з відкритим кодом, призначений для створення веб застосунків.

NgRx — фреймворк, призначений для керування станом застосунку, що підтримує парадигму реактивного програмування.

RxJS — бібліотека для складання асинхронних програм на основі подій.

MapStruct — генератор програмного коду, який значно спрощує реалізацію відображень між типами об’єктів Java.

Boilerplate-код — це розділи програмного коду, які повинні бути включені в багато місць програми, без змін узагалі або з незначними змінами.

Lombok — це проект з відкритим кодом, який зменшує кількість boilerplate коду у мові програмування Java, використовуючи анотації.

Spring Framework — це програмний каркас (фреймворк) з відкритим кодом та контейнери з підтримкою інверсії управління для платформи Java.

Typescript — мова програмування, представлена Microsoft восени 2012; позиціонується як засіб розробки веб-застосунків, що розширює можливості JavaScript.

REST (Representational State Transfer) — підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів.

JSON (JavaScript Object Notation) — це текстовий формат обміну даними між комп'ютерами.

JWT (JSON Web Token) — це стандарт токена доступу на основі JSON, стандартизованого в RFC 7519. Використовується для верифікації тверджень.

PWA (Progressive Web Application) — технологія веб-розробки, що візуально та функціонально трансформує сайт у мобільний застосунок(мобільний застосунок у браузері). Іншими словами, створений за технологією PWA, застосунок є гібридом веб- та мобільного застосунку. Створюється за допомогою можливостей що надають сучасні браузери, але при цьому його використання нагадує використання мобільного застосунку.

MVC (Model-View-Controller) — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

POJO (Plain Old Java Object) — простий Java-об'єкт, не успадкований від якогось специфічного об'єкта і який не реалізує жодних службових інтерфейсів окрім тих, які потрібні для бізнес-моделі.

State Management – термін, що позначає управління станом елементів керування інтерфейсом користувача, таких як текстові поля, кнопки ОК, радіо кнопки тощо у графічному інтерфейсі користувача.

Redux – шаблон проектування, що описує підхід до керування станом застосунку.

BCrypt — адаптивна криптографічна функція формування ключа, що використовується для безпечного зберігання паролів.

DigitalOcean — американський провайдер хмарних інфраструктур.

Дроплет — термін, що використовується компанією DigitalOcean для позначення віртуальних серверів, що надає компанія.

Learnability — один з атрибутів якості програмного забезпечення (згідно стандарту ISO 9126:2001), показник, зворотний зусиллям, що витрачається користувачами на навчання роботі з ПЗ.

Accessibility — можливість для маломобільних груп населення та зокрема осіб з особливими потребами отримувати доступ до певних продуктів, пристроїв, послуг, чи якогось середовища.

TestLodge — система управління тестуванням, що має веб-інтерфейс.

JAR (Java Archive) - це формат файлу пакету, який зазвичай використовується для об'єднання багатьох файлів класів мови програмування Java та пов'язаних з ними метаданих та ресурсів (текст, зображення тощо) в один файл для подальшого розповсюдження.

JRE – середовище, що надає бібліотеки, віртуальну машину Java та інші компоненти для запуску аплетів та програм, написаних мовою програмування Java.

PAAS – модель надання хмарних обчислень, при якій споживач отримує доступ до використання інформаційно-технологічних платформ: операційних систем, систем управління базами даних, зв'язного програмного забезпечення, засобів розробки і тестування розміщених у хмарних провайдерах.

Heroku – хмарна PaaS-платформа, що підтримує ряд мов програмування.

SSL-сертифікат – цифровий документ, який є одним із засобів підтвердження відкритого (публічного) ключа приналежності його власникові у моделі взаємодії протоколу SSL.

Let’s Encrypt – центр сертифікації, що надає безкоштовні криптографічні сертифікати X. 509 для TLS-шифрування (HTTPS) строком на 3 місці.

Firebase Hosting – це статичний та динамічний веб-хостинг, що підтримує хостинг статичних файлів, таких як CSS, HTML, JavaScript.

ВСТУП

У сучасному бурхливому світі щоденна активність людини складається з великої кількості подій. Життя стрімко набирає оберти і кількість подій, що відбуваються з людиною протягом дня стрімко зросла, в порівнянні з життям людей у попередньому столітті.

З ростом кількості подій, очевидно, зросла і складність процесу керування та управління тим, що відбувається з людиною. У відповідь на такі зміни зросла і складність інструментів, що їх використовують люди для управління якістю власного життя. Розглядаючи управління якістю власного життя як частковий випадок загальної проблеми управління якістю будь-чого (проблеми та підходи до їх вирішення розглядаються в рамках такої дисципліни як менеджмент), можна провести аналогію з таким інструментом менеджменту як “Цикл Шухарта-Демінга” (Plan-Do-Check-Act).

Цикл Шухарта-Демінга (тут і надалі PDCA) складається з 4-ох кроків, що ітеративно повторюються [1]:

- планування – встановлення цілей і процесів, необхідних для досягнення цілей;
- виконання – виконання запланованих робіт;
- перевірка – збір інформації та контроль результату на основі ключових показників ефективності (KPI);
- вплив – вжиття заходів, щодо усунення причин відхилення від запланованого результату, зміни в плануванні та розподілі ресурсів.

В рамках даного проекту будуть розглянуті інструменти, що забезпечують та спрощують виконання 3-ого кроку PDCA (перевірка). Наразі, люди стали активно впроваджувати аналіз та отримання зворотнього зв'язку в своє життя.

Підтвердженням цього є великий попит на різноманітні товари-трекери:

- фітнес-браслети;
- смарт-ваги;
- пристрої для моніторингу якості сну.

Подібні товари направлені на автоматичне зберігання базової, “низькорівневої” інформації про життя людини: активність протягом дня, робота серця (частота серцевих скорочень), характеристики сну(тривалість швидкої та повільної фаз), склад тіла(вимірювання кількості жирової, м’язової та кісткової маси тіла засобами біоімпедансометрії). Збір даних, як було зазначено вище, є однією із складових кроку “перевірка” PDCA. Але, подібна інформація, в силу її орієнтації на фізичні характеристики людини, придатна в основному для покращення фізичного здоров’я людини. Для більш глобальних цілей, очевидно, потрібні більш “високорівневі” інструменти.

Станом на сьогодні, наука та інженерія не здатні забезпечити можливість збору подібної високорівневої інформації в автоматичному режимі, без необхідності безпосередньої участі людини. Хоча, робота в цьому напрямку й ведеться (розробка нейрокомп’ютерних інтерфейсів), наразі серійного масштабу не досягнуто. Тому, на сьогодні, для того, збирати подібну інформацію люди, яким це цікаво, змушені робити це вручну.

Другою складовою етапу “перевірка” циклу PDCA є аналіз даних. Класичним підходом, до аналізу даних (включаючи і ручний аналіз) є інструменти математичної статистики. Такими засобами є:

- представлення даних у наочному, зручному для сприйняття мозком вигляді – різноманітні діаграми;
- структурування інформації – наприклад, класифікація інформації, організація інформації в ієрархічні структури;
- обчислення над масивом даних різних агрегатних функцій, наприклад, середнє значення, сума, мода, медіана.

В рамках даної роботи розглянуті підходи до побудови програмного забезпечення, що спрощують розв’язання вищезазначеної задачі. Задачею представленої роботи є створення системи для надання користувачам можливості зручно та структуровано зберігати власний життєвий досвід та аналізувати його. Під зручністю в даному контексті розуміється сучасний дизайн (UI, UX) та можливість отримання доступу онлайн, з широкого кола пристроїв, як мобільних так і стаціонарних.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

1.1.1 Дані та знання

Багаторазове зростання обсягів збереження даних в останні роки, темпи якого постійно збільшуються та зростаюче розмаїття збережуваної та оброблюваної інформації істотно ускладнюють задачу управління та використання накопиченого досвіду. В різні історичні періоди життя людства ця задача розв'язувалась різними способами. Одним з найпростіших способів, що його використовували люди, було ведення записів про події у хронологічному порядку. При цьому, для кожного запису зазначалась дата, до якої ця інформація відноситься. Подібні документи, залежно від характеру інформації, що в них описується, відомі як “літопис” або “щоденник особи”.

Відомо, що з появою комп’ютерної техніки сформувались два основні шляхи її використання. Перший спосіб - застосування обчислювальної техніки для виконання чисельних розрахунків, які дуже довго або взагалі неможливо здійснювати вручну. Розвиток цього напряму сприяв активному розвитку методів чисельних методів пошуку розв’язків складних задач математики, розвитку класу мов програмування, орієнтованих на зручний запис чисельних алгоритмів [2].

Другий спосіб – використання обчислювальної техніки в автоматизованих інформаційних системах. Інформаційна система, у широкому розумінні цього поняття, - це сукупність організаційних і технічних засобів для збереження та обробки інформації з метою забезпечення інформаційних потреб користувачів[3].

В рамках даного дипломного проекту розглядається, другий спосіб використання обчислювальної техніки – з метою забезпечити інформаційні

потреби користувача. Розділ інформатики, пов’язаний з розробкою систем, що забезпечують отримання, представлення, структурування і використання знань називається “Інженерія знань”.

ІЗ — розділ інженерії, направлений на впровадження знань в комп'ютерні системи для вирішення комплексів завдань, що зазвичай вимагають участі людського досвіду [4]. Одним з інструментів, що використовуються для цього, є організація накопиченого досвіду у спеціального виду бази даних. Такі бази даних називаються базами знань.

База знань - це особливого роду база даних, розроблена для управління знаннями (метаданими), тобто збором, зберіганням, пошуком і видачею знань [5].

Важливо розуміти, що поняття “дані” та “знання” не є тотожними. В контексті баз знань під поняттям “знання” розуміють певні закономірності, які були відкриті в шляхом професійної або трудової діяльності людей, що дозволяють робити постановку задач та вирішувати їх. Під поняттям “дані” розуміють окремі твердження, що описують об’єкти, процеси, явища предметної області, а також їх властивості. Таким чином, дані – це результат спостережень, тобто інформація, отримана емпіричним шляхом. Знання в свою чергу, засновані на даних. Знання отримують з даних, як результат процесу розумової діяльності людини, направленого на узагальнення досвіду, отриманого в результаті практичної діяльності. Згідно іншої точки зору, знання – це гарно організовані дані або метадані.

При обробці даних відбувається їх трансформація. В загальному вигляді, порядок трансформації даних можна представити у вигляді наступної послідовності станів:

- дані, як результат вимірювання або сприйняття побаченого;
- інформація на матеріальних носіях;
- моделі (структури) даних у вигляді діаграм, графіків;
- дані в комп’ютері на машинних носіях інформації.

Таким чином, ключем до переходу від даних до знань є процес структурування. Структурування інформації представляє собою процес виділення серед інформації окремих сутностей, групування подібних сутностей. В інформатиці, формалізоване представлення знань про певну предметну область, у вигляді, придатному для автоматизованої обробки, називається онтологією.

1.1.2 Формалізація представлення знань

Важливо розуміти, що онтологія є загальним поняттям. Онтологія визначає лише спосіб формалізації. В силу цього, може бути декілька різних способів формалізувати певну предметну область, а отже і декілька різних онтологій однієї предметної області. Ядром онтології неодмінно є деяка концепція предметної області. Найчастіше, така концепція виражається за допомогою визначення базових об’єктів (сутностей, категорій, атрибутів) та відношень між ними.

Зазвичай, онтології містять класи (поняття), екземпляри цих класів, їхні атрибути (властивості) та значення цих властивостей. Крім того, онтологія може містити певні обмеження на використання класів та їх відношень [6].

Отже, онтологія забезпечує структурування даних предметної області. Таким чином формується метамодель. Метамодель в інформатиці — модель, що описує іншу модель. Постає задача збереження структурованих даних на машинних носіях інформації.

Сучасним способом збереження даних є організація даних у вигляді бази даних. Задачу управління створенням та використанням баз даних вирішують використанням такого класу програмного забезпечення як СКБД (Система Керування Базами Даних). Залежно від використаної в СКБД моделі даних, розрізняють різні види СКБД – реляційні, ієрархічні, мережеві і т.д.

Однак, незалежно від обраного виду СКБД (реляційної, ієрархічної) постає завдання конвертації онтологічної моделі предметної області у модель збереження та організації даних СКБД. Пряме використання моделі організації даних СКБД найчастіше неможливе в силу наступних відмінностей баз даних від баз знань:

Таблиця 1.1 – Порівняння баз даних та баз знань

Бази даних	**Бази знань**
Оперують однотипними даними	Можуть зберігати різнотипні дані
Модель організації даних чітко визначена	Модель організації даних відкрита

Класичною методикою проектування баз даних є створення окремої таблиці для кожної описуваної моделлю даних сутності. Такий підхід придатний для БД з малою кількістю записів і при низькій складності моделі даних. Однак внесення змін до структури таблиць достатньо складна операція.

Розв’язком цієї проблеми є використання підходу, що полягає у відображенні метамоделі у реляційну модель бази даних. Засновником метамоделювання став Анатолій Тенцер. Він описав п'ять основних тез, на яких може бути побудована метамодель у реляційній базі даних [7]:

- кожна сутність, інформація про яку зберігається в БД, - це об'єкт;
- кожен об'єкт є унікальним у межах БД і має унікальний ідентифікатор;
- об'єкт має властивості (строкові, числові, тимчасові, перелічувані), які описують атрибути сутності;
- об'єкти можуть бути пов'язані між собою довільним чином;
- об'єкт може бути сховищем. У цьому випадку допускається зберігання в ньому інших об'єктів.

Така структура БД дозволяє описувати структуру майже будь якої предметної області.

Таким чином, використовуючи відображення метамоделі у реляційну модель можна реалізувати зберігання знань у реляційних базах даних. Загалом, таке відображення дозволяє зберігати базу знань у форматі реляційної бази даних, що дозволить зберігати у реляційні базі даних концептуально різнотипні сутності.

1.1.3 Моніторинг та аналіз

Розглянемо одне з формальних визначень поняття моніторинг. Моніторинг – це система постійного спостереження за явищами та процесами. В рамках системи спостереження відбувається оцінка, контроль та керування станом об’єкту в залежності від впливу певних факторів.

Інакше кажучи, моніторинг - це стандартизоване спостереження за процесом і його результатами, що дозволяє створювати історію стану об'єкта в часі, кількісно оцінювати зміну об’єктів, визначати і прогнозувати напрямки їх розвитку. Основна мета створення системи моніторингу - підвищення якості об’єкту або процесу.

Основними складовими моніторингу є:

- об’єкти моніторингу;
- комплекс показників якості;
- засоби накопичування інформації;
- методики аналізу, переробки та інтерпретації інформації.

Одним з ефективних способів ручного аналізу даних є їх візуалізація. Візуалізація даних – представлення даних у графічному вигляді. Вона передбачає відтворення даних у графічному вигляді таким чином, що робить очевидним для спостерігача наявні зв’язки між сутностями, що зображуються.

Ефективна візуалізація допомагає користувачам аналізувати та міркувати про дані та закономірності. Це робить складні дані більш доступними, зрозумілими та придатними до використання. Користувачі можуть мати конкретні аналітичні завдання, такі як порівняння чи розуміння причинності, а принцип розробки графіки (тобто демонстрація порівняння чи виявлення причинності) вже відповідає цьому завданню. Таблиці, як правило, використовуються там, де користувачам потрібні конкретні вимірювання, тоді як діаграми різних типів використовуються для відображення шаблонів чи зв'язків наявних в даних для однієї або декількох змінних [8].

1.2 Змістовний опис і аналіз предметної області

У розділі 1.1 було розглянуто теоретичні засади і принципи, на яких будуються інформаційні системи для управління знаннями. Ключовим фактором, що впливає на ефективність та якість результатів роботи таких систем, як було показано вище, є можливість гнучкого структурування інформації.

Оскільки система моніторингу життєвих подій має передусім вирішувати проблему збереження та аналізу життєвого досвіду(з метою отримання нових знань), то основною вимогою до такого програмного забезпечення є можливість гнучко структурувати інформацію.

Типовим підходом до структурування сутностей є класифікація. Класифікація передбачає поділ сутностей на окремі групи(класи) за певними ознаками. Однак, очевидно, що такий підхід має свої обмеження – в силу багатомірності сутностей реального світу, розділення їх на 1 набір класів неможливий(аналогічно неможливо однозначно відобразити тривимірний простір на площину). Тому, доцільно використовувати більшу кількість вимірів, наприклад використання декількох наборів класів. Наприклад, кожна сутність може бути віднесена до більше ніж одного класу. Такий підхід

відомий як тегування – з кожною сутністю асоціюється певний набір тегів(класів).

Іншою, не менш важливою характеристикою подібного програмного забезпечення є можливість візуалізації даних, а саме – представлення даних у вигляді графіків та діаграм.

1.3 Аналіз успішних IT-проектів

1.3.1 Аналіз відомих технічних рішень

Відповідно до технічного завдання проекту, основні властивості застосунку представлені наступним переліком:

- структуроване збереження інформації про подію;
- представлення інформації у зручному для аналізу вигляді;
- доступ онлайн зі стаціонарних та мобільних пристроїв.

В рамках аналізу відомих технічних рішень було розглянуто відомі індустріальні підходи до реалізації вищезазначеного переліку вимог.

Загалом, задача структурованого збереження інформації, має два концептуальні підходи.

Класичним та найбільш популярним підходом є побудова ER-моделі предметної області, на основі чітко визначених на етапі аналізу предметної області сутностей. При такому підході виділяються значущі сутності предметної області (а також їх атрибути) і зв’язки між сутностями. Далі, побудовану ER-модель перетворюють на конкретну схему бази даних. Розглянемо приклад перетворення ER-моделі на схему реляційної БД. При такому підході, кожна сутність ER-моделі представляється у вигляді окремої таблиці БД, а кожен атрибут сутності представляється у вигляді стовпця відповідної таблиці. Подібний підхід має суттєвий (для розглянутої в рамках даного дипломного проекту задачі) недолік – модель предметної області жорстко фіксується на етапі проектування БД, що призводить до неможливості заносити у БД інформацію про сутності, структура яких

відрізняється від структури сутностей ER-моделі. Хоча такий підхід і надає певні можливості структурування інформації, достатньої (для задачі, що розглядається в цій роботі) гнучкості структурування інформації він не забезпечує.

Іншим підходом, є використання метамоделей. В загальному випадку, метамодель описує структуру абстрактної предметної області. В рамках такого підходу обирається певна онтологія, на основі якої і будується структура БД. У найпростішому випадку побудована база даних буде складатись з 4 основних таблиць:

- таблиця типів – кожен рядок описує тип однієї сутності предметної області;
- таблиця атрибутів – кожен рядок описує атрибут сутності;
- таблиця об’єктів – кожен рядок описує конкретний екземпляр конкретного типу сутностей предметної області;
- таблиця параметрів – кожен рядок описує конкретне значення певного атрибуту об’єкту.

Схема розглянутої в прикладі БД наводиться нижче.

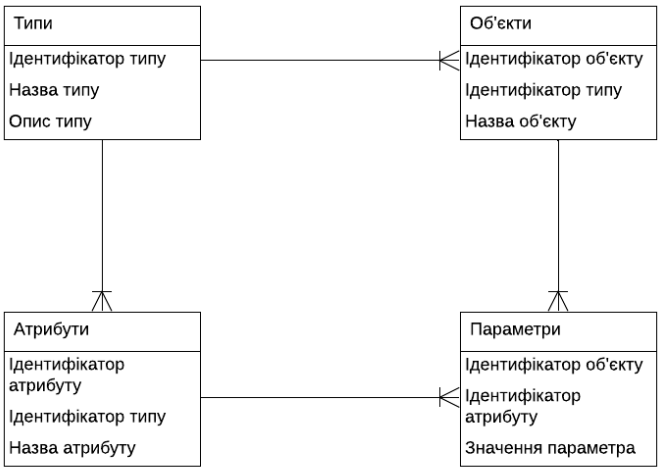


Рисунок 1.1 – Схема БД, що основана на метамоделі

Головною перевагою такого підходу є можливість додавання нових сутностей без необхідності перебудовувати схему БД. Завдяки цій

властивості, даний підхід дозволяє досягнути гнучкості при структуруванні інформації.

1.3.2 Аналіз відомих програмних продуктів

В ході дослідження було встановлено перелік основних програмних продуктів, що мають схожі функціональні можливості. Нижче наводиться перелік таких програмних продуктів з коротким описом переваг та недоліків.

RedNotebook – застосунок, що дозволяє вести журнал щоденних подій. Представляє собою прикладне програмне забезпечення, що потребує попереднього встановлення на комп'ютер. Має можливість пошуку записів. Структурування забезпечується тільки тегуванням. RedNotebook – типовий представник "Diary Software".

Серед переваг можна виділити:

- мультиплатформність (Windows, Linux, MacOS);
- простоту використання.

Недоліками є:

- необхідність попереднього встановлення;
- слабка структурованість інформації;
- відсутні інструменти аналізу.

YourDiary – мобільний застосунок, що дозволяє зберігати інформацію про події. Програмне забезпечення такого типу також відноситься до "Diary Software". Потребує попереднього встановлення на мобільний пристрій.

Серед переваг можна виділити:

- можливість поділитися інформацією про подію з іншим користувачем;
- можливість експорту даних;
- можливість побудови кругових діаграм на основі агрегованої інформації про події.

Недоліками є:

- відсутня можливість доступу зі стаціонарного комп’ютера;
- немає можливості збереження пов’язаних з подією файлів;
- обмежені статистичні можливості – присутня тільки можливість будувати кругові діаграми, що показують відсоток подій за кожним тегом;
- недостатня структурованість інформації.

Файлові сховища (GoogleDrive, Dropbox, NextCloud) – веб-застосунки, що надають користувачу можливість зберігати файли будь-якого типу та розміру. За бажанням, користувач може вносити інформацію про свій життєвий досвід у текстові файли, організовувати їх ієрархічно та зберігати поряд з описом події пов’язані з нею документи(фото-, відео- матеріали, тощо). При такому способі використання, структурування забезпечується можливостями формувати ієрархії папок.

До переваг відноситься:

- простота використання;
- можливість ієрархічного структурування;
- доступ як зі стаціонарних так і мобільних пристроїв;
- немає необхідності встановлювати програмне забезпечення, оскільки присутній веб-інтерфейс.

Недоліками є:

- відсутність аналізу;
- недостатня структурованість.

Flowlu – веб-платформа, призначена для спрощення та автоматизації бізнес-процесів підприємства. Має широкий перелік функцій, одна з яких – створення бази знань. Дозволяє створювати записи, структурувати їх за допомогою тегів, прикріпляти до них документи (у вигляді прикріплених файлів). Перш за все, функціонал такої бази знань орієнтований на забезпечення потреб бізнесу зберігати інформацію про організацію внутрішніх процесів підприємства. Однак, може бути використаний як

простий розв’язок проблеми, що розглядається в рамках даного дипломного проекту.

Перевагами є:

- можливість сумісного з іншими користувачами внесення/модифікації інформації;
- можливість прикріплювати файли до записів;
- можливість структурування за допомогою тегів.

Недоліками є:

- недостатня структурованість;
- відсутність аналізу.

Підсумовуючи результати аналізу відомих програмних продуктів, зроблено висновок про те, що на ринку присутні продукти, що за своїми функціональними можливостями можуть бути використані для вирішення задачі, що розглядається в рамках даного дипломного проекту, їх функціонал має 2 основних недоліки: недостатня структурованість збережуваної інформації та обмежені (або взагалі відсутні) засоби аналізу.

1.4 Аналіз вимог до програмного забезпечення

Визначимо перелік ролей користувачів системи. Кожен користувач системи може мати одну з наведеного нижче переліку ролей:

- авторизований користувач;
- неавторизований користувач.

Після успішного завершення процесу входу в систему неавторизований користувач набуває ролі авторизованого користувача системі, що дає йому можливість використовувати всі функції системи.

Авторизований користувач може створювати такі сутності у системі:

- тег;
- метрика;

- категорія;
- подія.

Кожна створена окремим користувачем сутність ізольована від створених іншим користувачем сутностей. Тобто, всі об’єкти, що були створені одним користувачем недоступні для перегляду, редагування або видалення іншим користувачем системи.

1.4.1 Розроблення функціональних вимог

В процесі проведення аналізу вимог до застосунку було виділено та поділено вимоги на функціональні та нефункціональні. Функціональні вимоги – це вимоги до програмного забезпечення, що описують роботу системи, її поведінку: внесення даних, обробка даних та інші специфічні операції, які має робити система. У програмному забезпеченні для моніторингу життєвих подій було виділено функціональні вимоги, що описані в таблицях нижче. У рядку таблиці “Перелік ролей” зазначено перелік ролей, до яких відноситься дана функціональна вимога.

Таблиця 1.2 – Опис функціональної вимоги REQ001

Номер	REQ001
Назва	Реєстрація в системі
Опис	Користувач вносить інформацію про себе у систему
Перелік ролей	Неавторизований користувач

Таблиця 1.3 – Опис функціональної вимоги REQ002

Номер	REQ002
Назва	Вхід в систему
Опис	Користувач вводить валідний логін та пароль, після чого набуває ролі “Авторизований користувач”
Перелік ролей	Неавторизований користувач

Таблиця 1.4 – Опис функціональної вимоги REQ003

Номер	REQ003
Назва	Створення тегу
Опис	Користувач може створити тег у системі. Тег повинен містити таку інформацію: - назва тегу; - опис тегу; - колір тегу.
Перелік ролей	Авторизований користувач

Таблиця 1.5 – Опис функціональної вимоги REQ004

Номер	REQ004
Назва	Створення метрики
Опис	Користувач може створити метрику у системі. Метрика повинна містити таку інформацію: - назва метрики; - опис метрики; - тип метрики (числовий або номінальний).
Перелік ролей	Авторизований користувач

Таблиця 1.6 – Опис функціональної вимоги REQ005

Номер	REQ005
Назва	Створення категорії події
Опис	Користувач може створити категорію у системі. Категорія повинна містити таку інформацію: - назва категорії; - опис категорії; - колір категорії; - перелік метрик, що відносяться до даної категорії.

Таблиця 1.7 – Опис функціональної вимоги REQ006

Номер	REQ006
Назва	Створення події
Опис	Користувач може створити подію у системі. Подія повинна місити таку інформацію: - назва події; - опис події; - дату та час події; - категорію події; - перелік значень кожної метрики, що належить обраній категорії події. Опціонально, користувач, при створенні події, може визначити перелік тегів, до яких належить ця подія.
Перелік ролей	Авторизований користувач

Таблиця 1.8 – Опис функціональної вимоги REQ007

Номер	REQ006
Назва	Пошук події
Опис	Користувач може знайти події, що відповідають пошуковому запиту. Критерій відповідності події пошуковому запиту – текст запиту міститься у назві або описі події
Перелік ролей	Авторизований користувач

Таблиця 1.9 – Опис функціональної вимоги REQ008

Номер	REQ006
Назва	Пошук метрики
Опис	Користувач може знайти метрики, що відповідають пошуковому запиту. Критерій відповідності метрики пошуковому запиту – текст запиту міститься у назві або описі метрики
Перелік ролей	Авторизований користувач

Таблиця 1.10 – Опис функціональної вимоги REQ009

Номер	REQ006
Назва	Пошук тегу
Опис	Користувач може знайти теги, що відповідають пошуковому запиту. Критерій відповідності тегу пошуковому запиту – текст запиту міститься у назві або описі тегу.
Перелік ролей	Авторизований користувач

Таблиця 1.11 – Опис функціональної вимоги REQ010

Номер	REQ006
Назва	Пошук категорії
Опис	Користувач може знайти категорії, що відповідають пошуковому запиту. Критерій відповідності категорії пошуковому запиту – текст запиту міститься у назві або описі категорії.
Перелік ролей	Авторизований користувач

Таблиця 1.12 – Опис функціональної вимоги REQ011

Номер	REQ006
Назва	Перегляд статистики за категоріями
Опис	Користувач може переглянути кругову діаграму, що показує кількість подій у системі, за кожною категорією за визначений проміжок часу.
Перелік ролей	Авторизований користувач

Таблиця 1.13 – Опис функціональної вимоги REQ012

Номер	REQ006
Назва	Перегляд статистики за тегами
Опис	Користувач може переглянути кругову діаграму, що показує кількість подій у системі, за кожним тегом за визначений проміжок часу.
Перелік ролей	Авторизований користувач

Таблиця 1.14 – Опис функціональної вимоги REQ013

Номер	REQ006
Назва	Перегляд статистики за метриками
Опис	Користувач може переглянути лінійний графік, що показує динаміку зміни значень метрик у часі. Перелік метрик, що показані на графіку визначається користувачем. Значення показуються з кроком часу у 1 місяць. Значення метрики за кожен конкретний місяць агрегуються відповідно до обраної користувачем агрегуючої функції. За потреби, користувач може привести значення на графіку до безрозмірних величин, шляхом нормалізації даних.
Перелік ролей	Авторизований користувач

Таблиця 1.15 – Опис функціональної вимоги REQ012

Номер	REQ006
Назва	Ізоляція створених сутностей
Опис	Кожен користувач має можливість переглядати сутності(теги, метрики, категорії, події), що були створені ним. Сутності, що були створені іншим користувачем недоступні для перегляду, редагування або видалення. Інформації, що міститься у сутностях іншого користувача, не має відображатися в тому числі і у статистичних звітах(графіки, діаграми)
Перелік ролей	Авторизований користувач

На основі сформульованих функціональних вимог була побудована структурна схема використання. Побудована схема наведена у документі КПІ.ІП-6326.045420.07.99.СС “Схема структурна варіантів використання системи”.

1.4.2 Розроблення нефункціональних вимог

Була проведена розробка нефукціональних вимог до програмного забезпечення. Виділені нефунціональні вимоги описані в таблицях 1.16 – 1.20.

Таблиця 1.16 – Опис нефункціональної вимоги NFR001

Номер	NFR001
Назва	Підтримка версій мови програмування Java
Опис	Серверна частина програмного забезпечення повинна бути сумісна з JRE 8.

Таблиця 1.17 – Опис нефункціональної вимоги NFR002

Номер	NFR002
Назва	Працездатність
Опис	Веб-застосунок повинен працювати тільки якщо комп’ютер має доступ до мережі Інтернет.

Таблиця 1.18 – Опис нефункціональної вимоги NFR003

Номер	NFR003
Назва	Швидкодія
Опис	Система повинна реагувати на дії користувача з затримкою, що не перевищує 5 секунд.

Таблиця 1.19 – Опис нефункціональної вимоги NFR004

Номер	NFR004
Назва	Зручність і простота
Опис	Клієнтська частина програмного забезпечення повинна бути зручною та інтуїтивно зрозумілою для користувачів.

Таблиця 1.20 – Опис нефункціональної вимоги NFR005

Номер	NFR005
Назва	Кросплатформенність
Опис	Клієнтська частина програмного забезпечення має коректно функціонувати як на стаціонарних так і на мобільних пристроях

1.4.3 Постановка комплексу завдань модулю

Згідно з розробленими функціональними та нефункціональними вимогами встановимо комплекс завдань, що має вирішити розроблюваний програмний модуль:

– надати можливість користувачу створювати обліковий запис;

– надати можливість користувачу визначати моделі подій;

– надати можливість користувачу вносити інформацію про події;

– надати можливість користувачу виконувати пошук подій;

– надати можливість користувачу переглядати статистичні звіти.

Комплекс завдань модулю має вирішувати усі задачі системи.

1.5 Висновки по розділу

У розділі 1 була зроблена постановка задачі, оглянуто та висвітлено основні проблеми та наявні методи реалізації подібних задач.

Були розглянуті успішні IT-проекти, що вирішують схожий набір задач. Було виконано порівняння з системою, що розробляється та виявлені недоліки та переваги системи у порівнянні з можливими конкурентами. В результаті порівняння було з’ясовано, що потенційні конкуренти не надають достатніх засобів структурування та аналізу інформації. Розглянувши відомі технічні рішення та підходи було з’ясовано, що існують способи виправити розглянуті вище недоліки.

Також була розроблена система функціональних та нефункціональних вимог. Окрім цього, був поставлений комплекс завдань для розроблюваного модулю згідно з системою функціональних та нефункіональних вимог.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Базуючись на вимогах до програмного забезпечення для моніторингу життєвих подій було проведено моделювання та аналіз програмного забезпечення. Для проектування бізнес-процесів було обрано методологію BPMN. У ході аналізу та моделювання програмного забезпечення для моніторингу життєвих подій було виділено декілька основних бізнес-процесів:

- вхід у систему (аутентифікація);
- реєстрація у системі(створення облікового запису користувача);
- маніпулювання структуруючими сутностями:
  - маніпулювання тегами;
  - маніпулювання метриками;
  - маніпулювання категоріями.
- маніпулювання подіями;
- перегляд статистичних даних.

Відповідні BPMN діаграми наведено нижче.

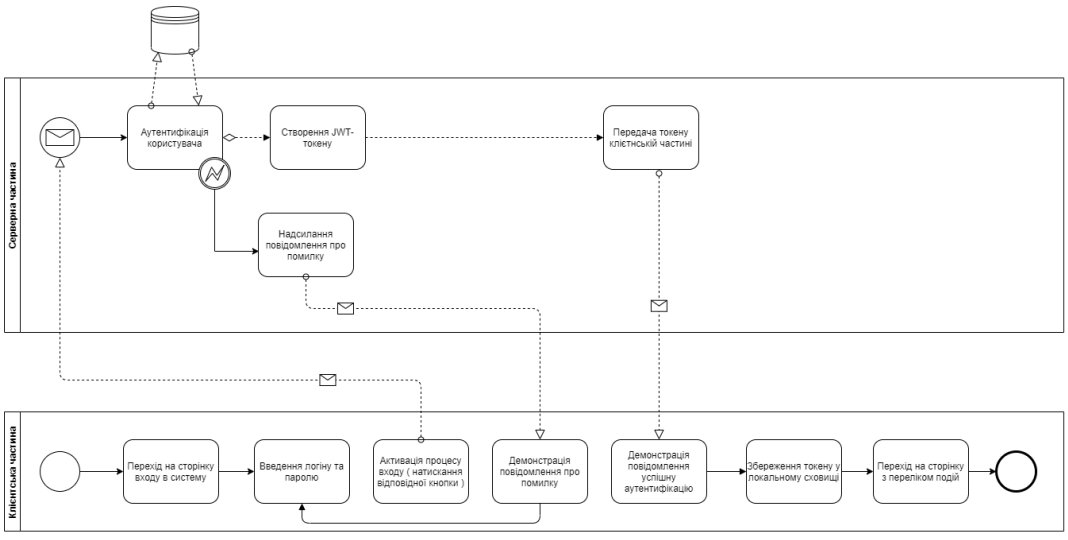


Рисунок 2.1 – Схема бізнес-процесу аутентифікації

Рисунок 2.2 – Схема бізнес-процесу реєстрації

Рисунок 2.3 – Схема бізнес-процесу маніпулювання тегами

Рисунок 2.4 – Схема бізнес-процесу маніпулювання метриками

Рисунок 2.5 – Схема бізнес-процесу маніпулювання категоріями

Рисунок 2.6 – Схема бізнес-процесу маніпулювання подіями

Рисунок 2.7 – Схема бізнес-процесу перегляду статистики за категоріями

Рисунок 2.8 – Схема бізнес-процесу перегляду статистики за тегами

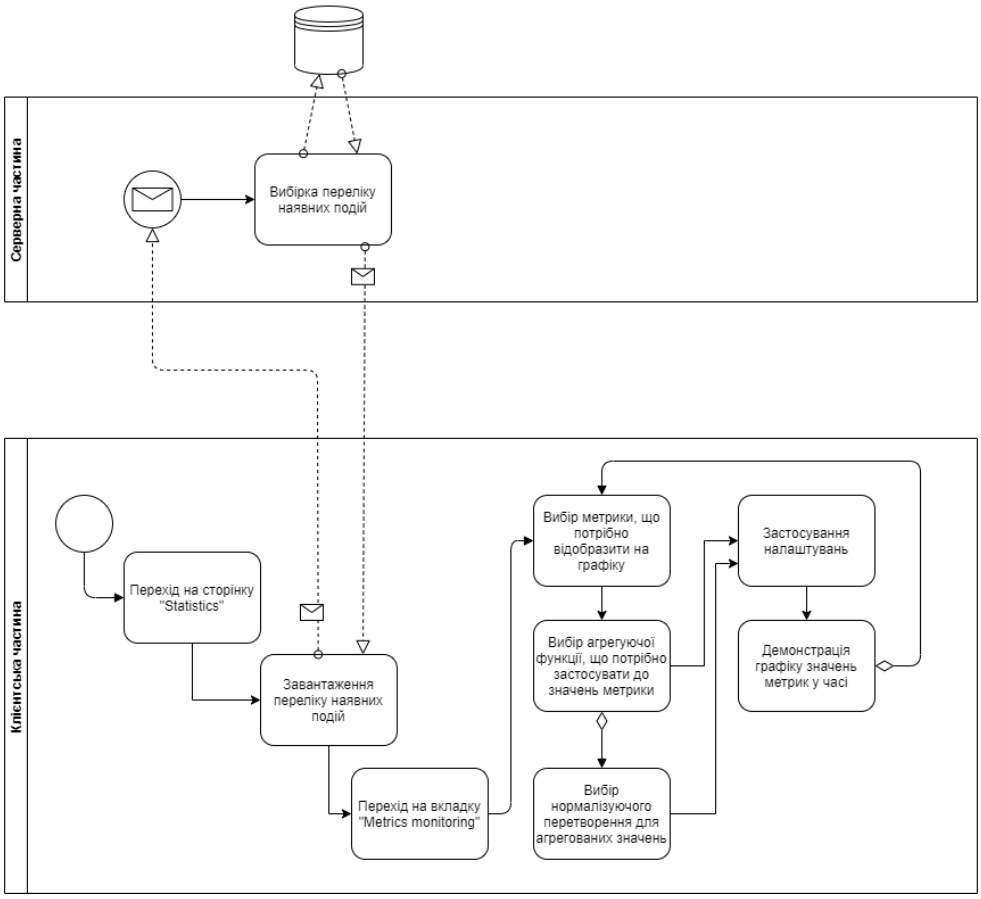


Рисунок 2.9 – Схема бізнес-процесу перегляду графіку значень метрик у часі

2.2 Архітектура програмного забезпечення

Засновуючись на вимогах, результатах аналізу та моделювання програмного забезпечення було прийнято рішення використати для реалізації рішення наступний перелік технологій.

Для розробки серверної частини:

- мову програмування Java;
- фреймворк загального призначення Spring;
- бібліотеку Lombok (генерація вихідних кодів на основі анотацій);
- бібліотеку MapStruct (для підтримки DTO);
- систему керування базами даних PostgreSQL.

Для розробки клієнтської частини:

- мову програмування Typescript;

- фреймворк Angular;
- бібліотеку Angular Material (надає широкий набір типових елементів графічного інтерфейсу, що відповідає сучасним вимогам до елементів UI);
- фреймворк NgRx (надає можливість створювати сховище застосунку та забезпечує роботу з ним у парадигмі реактивного програмування);
- бібліотеку RxJS (надає можливість оперувати даними у реактивній парадигмі програмування [9]).

Зазначені вище технології були обрані, оскільки вони надають всі необхідні для створення застосунку Biograph можливості(надають можливість задовольнити всі сформульовані у розділі 1 вимоги). Разом з тим, наразі такий набір технологій активно підтримується спільнотою розробників, що забезпечує можливість простого додавання необхідного функціоналу у майбутньому (оскільки з розвитком технологій та вимог до програмного забезпечення буде розвиватись і зазначений вище технологічний стек).

Задля забезпечення взаємодії клієнтської та серверної частини застосунку було обрано протокол HTTPS, який наразі є індустріальним стандартом для подібного формату програмних засобів. Серверна частина застосунку надає доступ до API, що задовольняє вимогам REST. Для серіалізації даних був обраний формат JSON. Щоб забезпечити захист від несанкціонованого доступу до API використовуються JWT-токени.

Задля забезпечення можливості зручного використання додатку на мобільних пристроях(смартфони, планшети) було прийнято рішення використовувати технологію PWA. Це дасть можливість створити веб застосунок, використання якого на мобільних пристроях нагадує використання мобільного застосунку. Для цього користувачу необхідно у

відкрити застосунок у браузері, що підтримує технологію PWA та встановити його мобільну версію.

2.2.1 Структура бази даних

На етапі проектування бази даних був використаний підхід відображення метамоделі у реляційну модель. Детальніше цей підхід був розглянутий у розділі 1. Схема побудованої бази даних наводиться у документі КПІ.ІП-6326.045420.07.99.СБД “Схема бази даних”. Список основних таблиць та їх опис наводиться у таблиці 2.1.

Таблиця 2.1 – Опис таблиць бази даних

Назва таблиці	Опис таблиці
TAGS	Таблиця, що містить у собі інформацію про наявні у системі теги. Кожен рядок містить інформацію про назву, опис, колір та дату створення тегу.
EVENTS	Таблиця, що містить інформацію про наявні у системі теги. Кожен рядок містить інформацію про назву, опис, час початку та завершення події, дату створення та останньої модифікації а також ідентифікатор категорії події
ATTACHMENTS	Таблиця, що містить інформацію про прикріплені до події файли. Кожен рядок містить інформацію про назву файлу, його опис, внутрішній ідентифікатор файлу у сховищі а також ідентифікатор події, до якої прикріплено цей файл.

Продовження таблиці 2.1

CATEGORIES	Таблиця, що містить інформацію про наявні у системі категорії подій. Кожен рядок містить інформацію про назву, колір, опис та дату створення категорії.
ATTRIBUTES	Таблиця, що містить інформацію про наявні в системі атрибути. В графічному інтерфейсі користувача об’єкти, що зберігаються у даній таблиці називаються метриками. В даному випадку ці поняття слід розуміти як синонімічні. Це зроблено для спрощення розуміння суті користувачем, незнайомим з термінологію метамоделювання.
CONSTRAINTS	Таблиця, що містить інформацію про тип та валідаційні обмеження значень атрибуту. Тип атрибуту визначається пов'язаним з ним об’єктом constraint. Реляційний зв’язок між сутностями таблиць CONSTRAINTS та ATTRIBUTES – один до одного.
PARAMETERS	Таблиця, що містить параметри(конкретні значення атрибутів) події. Кожен рядок містить посилання на подію та атрибут.

Продовження таблиці 2.1

USERS	Таблиця, що містить інформацію про облікові записи зареєстрованих у системі користувачів.
GRANTS	Таблиця, що містить інформацію про права доступу користувачів до окремих сутностей, що зберігаються у БД. Дана таблиця є точкою подальшого(не описаного в рамках технічного завдання) розширення функціоналу системи. Наявність даної таблиці надає можливість спільного доступу декількох користувачів системи до збережуваних даних(подій, категорій, тегів, тощо). Також, наявність такої таблиці значно спрощує подальшу реалізацію наскрізного шифрування даних.

Детальний опис полів таблиць наведено нижче:

Таблиця 2.2 – Опис полів таблиці Tags

Назва поля	Тип	Опис
tag_id	Числовий	Унікальний ідентифікатор тегу
name	Текстовий	Назва тегу
description	Текстовий	Опис тегу
color	Текстовий	Колір тегу у кольоровій схемі RGB
creation_time	Дата та час	Дата створення тегу

Таблиця 2.3 – Опис полів таблиці Events

Назва поля	Тип	Опис
event_id	Числовий	Унікальний ідентифікатор події
name	Текстовий	Назва події
description	Текстовий	Текстовий опис події
start_datetime	Дата та час	Дата та час початку події
end_datetime	Дата та час	Дата та час завершення події
last_modified_time	Дата та час	Дата останньої модифікації події
category_id	Числовий	Ідентифікатор категорії, до якої належить подія

Таблиця 2.4 – Опис полів таблиці Attachments

Назва поля	Тип	Опис
attachment_id	Числовий	Унікальний ідентифікатор прикріпленого файлу
file_name	Текстовий	Назва файлу, при додаванні користувачем
file_name_in_storage	Текстовий	Внутрішній ідентифікатор файлу у системі
description	Текстовий	Текстовий опис прикріпленого файлу
event_id	Числовий	Ідентифікатор події, до якої прикріплено файл

Таблиця 2.5 – Опис полів таблиці Categories

Назва поля	Тип	Опис
category_id	Числовий	Унікальний ідентифікатор категорії
name	Текстовий	Назва категорії
description	Текстовий	Текстовий опис категорії
color	Текстовий	Колір категорії, у кольоровій схемі RGB
creation_time	Дата та час	Дата та час створення категорії

Таблиця 2.6 – Опис полів таблиці Attributes

Назва поля	Тип	Опис
attribute_id	Числовий	Унікальний ідентифікатор атрибуту
name	Текстовий	Назва атрибуту
description	Текстовий	Текстовий опис атрибуту
attribute_type	Текстовий	Тип атрибуту
creation_time	Дата та час	Дата та час створення атрибуту
constraint_id	Числовий	Ідентифікатор сутності, що описує допустимі значення атрибуту

Таблиця 2.7 – Опис полів таблиці Constraints

Назва	Тип	Опис
id	Числовий	Унікальний ідентифікатор обмеження значень атрибуту
min	Числовий	Мінімальне значення атрибуту(включно)
max	Числовий	Максимальне значення атрибуту(не включно)
possible values	Текстовий	Перелік, розділених комою, допустимих значень атрибуту(тільки для номінальних атрибутів)

Таблиця 2.8 – Опис полів таблиці Parameters

Назва	Тип	Опис
parameter_id	Числовий	Унікальний ідентифікатор параметру
value	Текстовий	Значення параметра(для числових атрибутів – у десятковій системі)
attribute_id	Числовий	Ідентифікатор атрибуту, значення якого зберігається у параметрі
event_id	Числовий	Ідентифікатор події, до якої належить параметр

Таблиця 2.9 – Опис полів таблиці Users

Назва поля	Тип	Опис
user_id	Числовий	Унікальний ідентифікатор користувача
nickname	Текстовий	Псевдонім користувача
current_email	Текстовий	Актуальна адреса електронної пошти
email_confirmed	Логічний	Флаг, що показує чи було перевірено, що пошта належить користувачу
hash_function_type	Текстовий	Тип хеш-функції, що використовувалась для обчислення значення поля “password_hash”
is_compromised	Логічний	Флаг, що показує, чи був скомпрометований обліковий запис користувача
password_hash	Текстовий	Образ паролю користувача, що обчислено за допомогою хеш-функції
creation_time	Дата та час	Дата та час створення облікового запису користувача

Таблиця 2.10 – Опис полів таблиці Grants

Назва поля	Тип	Опис
grant_id	Числовий	Унікальний ідентифікатор права доступу
access_type	Текстовий	Тип права доступу. Допустимі значення типу: - Owner - Read_Only - Edit
attribute_id	Числовий	Ідентифікатор атрибуту, права доступу до якого описуються
category_id	Числовий	Ідентифікатор категорії, права доступу до якої описуються
event_id	Числовий	Ідентифікатор події, права доступу до якої описуються
tag_id	Числовий	Ідентифікатор тегу, права доступу до якого описуються
user_id	Числовий	Ідентифікатор користувача, якому надаються права доступу

2.2.2 Архітектура серверної частини

Серверна частина застосунку побудована за принципами архітектурного шаблону MVC. Цей шаблон описує поділ системи на три основні частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача [10].

Відповідно до шаблону MVC, модель даних представлена у застосунку у вигляді сервісів та POJO-об’єктів, що описують сутності.

Відокремлення моделі від інтерфейсу користувача забезпечуються за допомогою DTO. Цей шаблон пропонує відділити внутрішнє представлення сутностей у системі від формату API. Це досягається за допомогою опису спеціальних класів, що описують формат взаємодії з API. Процес конвертації DTO-об’єктів у внутрішнє представлення системи і назад забезпечується спеціальними сервісами – “DTO-мапперами”. Задля цього у даному проекті використана бібліотека MapStruct.

Модуль керування представлений спеціальними класами-контролерами. Ці класи призначені для отримання API-запитів та передачі результатів запиту користувачу API.

2.2.3 Архітектура клієнтської частини

Клієнтська частина застосунку заснована на компонентно-орієнтованому підході. Суть підходу полягає у виділенні окремих елементів програми (компонентів) що виконують конкретну функцію. В контексті розробки веб-інтерфейсів компонент найчастіше складається з 3 складових:

- опис логічної структури компоненту (найчастіше у HTML форматі);

- опис зовнішнього виду (найчастіше описується мовами стилів – CSS, LESS, тощо);
- опис логіки роботи компоненту (найчастіше описується за допомогою імперативних мов програмування).

В клієнтській частині застосунки активно використовуються інструменти реактивного програмування. Засоби реактивної парадигми програмування дозволяють досягти зручності використання застосунку користувачем за рахунок того, що будь-яка дія користувача не потребує оновлення вкладки браузера.

Керування станом застосунку(State management) реалізовано відповідно до шаблону проектування Redux [11]. Задля реалізації цього шаблону проектування використана бібліотека NgRx. Діаграма, що описує життєвий цикл керування станом застосунку наведена нижче.

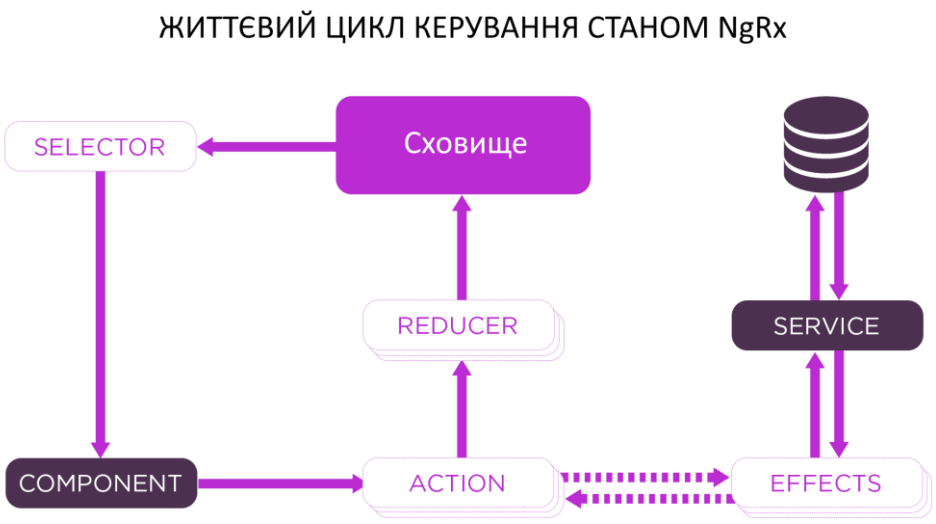


Рисунок 2.10 – Життєвий цикл керування станом NgRx

В рамках життєвого циклу виділяються специфічні для бібліотеки NgRx сутності. Нижче наведено їх перелік та призначення:

- action – інкапсулює в собі певну дію (наприклад, завантаження переліку подій);
- component – базова складова інтерфейсу (наприклад, компонента для відображення форми авторизації);

- reducer – змінює стан сховища на основі його попереднього стану та переданої в параметрах події (наприклад, додавання завантажених с серверної частини подій у сховище);
- effects – обробляє action та забезпечує застосунку доступ до шару сервісів. При такому підході, сервіси, надають доступ до REST API серверної частини застосунку;
- selector – функція, що дозволяє компоненту отримати поточне значення стану сховища.

2.2.4 Опис класів

Програмне забезпечення спроектовано з використанням об’єктно-орієнтованого підходу. Класи додатку структуровано за допомогою групування їх в окремі модулі(пакети). Загалом, виділено наступні програмні пакети:

- контролери;
- сервіси;
- класи, що описують предметну область(домен);
- репозиторії – забезпечують доступ до шару бази даних;
- утилітні класи – забезпечують виконання загальних функцій, не пов’язаних з предметною областю(обчислення криптографічних функції, генерація та валідація токенів доступу, математичні операції, тощо).

Класи-сервіси виконують також функцію конвертації DTO у внутрішні сутності системи.

Розбиття класів на окремі групи дозволяє розробнику краще орієнтуватись у текстах програмного коду та керувати складністю розроблюваного продукту. Перелік основних класів та їх функцій наведено в таблицях нижче.

Таблиця 2.11 – Опис класів-контролерів

Назва класу	Перелік функції
AttributeController	- Отримання переліку всіх метрик, які належать користувачу. - Створення метрики. - Видалення метрики. - Редагування метрики.
CategoryController	- Отримання переліку всіх категорій, які належать користувачу. - Створення категорії. - Видалення категорії. - Редагування категорії.
EventController	- Отримання переліку всіх подій, що належать користувачу. - Створення події. - Редагування події. - Видалення події.
TagController	- Отримання переліку всіх тегів, що належать користувачу. - Створення тегу. - Редагування тегу. - Видалення тегу.
AuthController	- Створення облікового запису користувача(реєстрація). - Авторизація користувача(генерація та отримання JWT-токену).

Таблиця 2.12 – Опис класів-сервісів

Назва класу	Перелік функцій
AttributeService	- Створення метрики. - Видалення метрики. - Пошук метрики за ідентифікатором. - Отримання переліку всіх метрик, що належать користувачу. - Перевірка права доступу користувача до метрики.
CategoryService	- Створення категорії. - Отримання переліку категорій, що належать користувачу. - Пошук категорії за ідентифікатором. - Видалення категорії. - Редагування категорії. - Перевірка права доступу користувача до категорії.
EventService	- Створення події. - Отримання переліку подій, що належать користувачу. - Редагування події. - Видалення події. - Перевірка права доступу користувача до події.

Продовження таблиці 2.12

GrantService	- Отримання переліку прав доступу користувача.
JwtUserDetailsService	- Отримання інформації про користувача, маючи адресу електронної пошти користувача.
ParameterService	- Перевірка відповідності значення параметра обмеженням на допустимі значення відповідного атрибуту.
AuthService	- Створення облікового запису користувача.
TagService	- Отримання переліку тегів, що належать користувачу. - Пошук тегу за ідентифікатором. - Редагування тегу. - Видалення тегу. - Перевірка прав доступу поточного користувача до тегу.
UserService	- Пошук облікового запису користувача за адресою його електронної пошти. - Пошук облікового запису користувача за її ідентифікатором.

Таблиця 2.13 – Опис класів, що представляють сутності предметної області

Назва класу	Сутність предметної області, що описується
Attachment	Прикріплений до події файл
Attribute	Метрика події
Category	Категорія події
Event	Подія
Grant	Право доступу до об’єкту системи
Parameter	Конкретне значення атрибуту певної події
Tag	Тег
User	Обліковий запис користувача

Таблиця 2.14 – Опис класів-репозиторіїв

Назва класу	Функції
AttributeRepository	- Створення метрики. - Пошук метрики за ідентифікатором. - Отримання переліку всіх категорій. - Видалення метрики. - Оновлення запису про метрику у БД.

Продовження таблиці 2.14

CategoryRepository	- Створення категорії. - Отримання переліку всіх категорій. - Оновлення запису про категорію у БД. - Видалення категорії.
EventRepository	- Отримання переліку всіх подій. - Створення запису про подію у БД. - Видалення запису про подію з БД. - Оновлення запису про подію у БД.
GrantRepository	- Отримання переліку всіх записів про права доступу користувача, що описують доступ до метрики. - Отримання переліку всіх записів про права доступу користувача, що описують доступ до тегу. - Отримання переліку всіх записів про права доступу користувача, що описують доступ до події. - Отримання переліку всіх записів про права доступу до категорії.

Продовження таблиці 2.14

TagRepository	- Створення запису про тег у БД. - Отримання переліку всіх записів про тег у БД. - Оновлення запису про тег у БД. - Видалення запису про тег з БД.
UserRepository	- Пошук облікового запису користувача за адресою електронної пошти. - Перевірка наявності облікового запису користувача, що містить зазначений псевдонім(nickname) користувача.

Таблиця 2.15 – Опис утилітних класів

Назва класу	Функції
JwtTokenUtil	- Отримання адреси електронної пошти власника JWT-токену. - Генерація JWT-токену. - Генерація випадкового ідентифікатору JWT-токену.

2.3 Аналіз безпеки даних

Програмне забезпечення, що розроблюється в рамках даного дипломного проекту передбачає захист даних від несанкціонованого доступу. Такий захист забезпечується механізмами авторизації та аутентифікації.

Для отримання доступу до даних застосунок вимагає аутентифікації. Під час входу користувача у систему, користувач мусить надати коректну пару логін/пароль. У разі успішного проходження аутентифікації клієнтська частина застосунку отримує JWT-токен, який і використовується для подальшої аутентифікації запитів.

З метою запобігання несанкціонованого доступу до системи паролі у БД не зберігаються у відкритому вигляді. У БД зберігаються хеші паролів, отримані з використання криптографічної функції BCrypt.

Система має механізми розділення даних користувачів(модель заснована на правах доступу). Кожен запит користувача до системи проходить процес авторизації. Якщо користувач намагається провести операцію над даними, доступу до яких йому заборонено, системи відхиляє такий запит.

Система спроектована таким чином, що передбачає можливість додавання функціоналу наскрізного шифрування даних(підхід, при якому всі вразливі до витоку дані шифруються та розшифровуються на клієнтській частині застосунку).

Захист комунікаційного каналу між клієнтською та серверною частинами, а також між серверною частиною та СУБД забезпечується криптографічним протоколом SSL.

Таким чином, вищезазначені підходи забезпечують два основних принципи інформаційної безпеки програмних засобів:

- конфіденційність – гарантія того, що дані не розголошуються неавторизованим суб’єктам. В рамках спроектованого продукту, конфіденційність забезпечується механізмами аутентифікації, авторизації та забезпеченням захищеного каналу зв’язку;
- цілісність – гарантія того, що дані захищені від неавторизованих модифікацій. В рамках спроектованого продукту, цілісність забезпечується механізмами авторизації та використанням моделі прав доступу.

2.4 Висновки по розділу

В даному розділі було проведено моделювання та аналіз програмного забезпечення для моніторингу життєвих подій. Окреслено архітектуру програмного забезпечення, яка буде використовуватися при розробці застосунку.

В рамках даного розділу була розроблена та описана структура БД.

Було проведено аналіз безпеки даних. В результаті аналізу показано, що спроектована система має функціональні можливості, для забезпечення безпечного зберігання та доступу до інформації, що зберігається у застосунку.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вступ

Задля проведення випробувань програмного продукту Biograph та визначення ступеня відповідності реальної поведінки програми очікуваній поведінці розроблено план тестування. Розглянутий у цьому розділі план тестування базується на міжнародному стандарті IEEE 829-2008.

Метою даного тестового плану є:

- визначити перелік інструментів, що будуть використовуватися в процесі тестування;
- визначити спосіб проведення тестів;
- визначити перелік функцій, що підлягають тестуванню, встановити очікувані терміни проведення тестування та сформувати вимоги до тестового середовища.

3.2 Об’єкти тестування

Перелік програмних продуктів, що будуть тестуватися, включає в себе клієнтську та серверну частину застосунку Biograph.

Клієнтська частина Biograph має бути протестована у браузерах:

- Google Chrome версії 81.0;
- Mozilla Firefox версії 76.0.1.

Серверна частина застосунку Biograph має бути протестована у середовищі, що задовольняє таким вимогам:

- операційна система Ubuntu Server версії 16.04;
- об’єм оперативної пам’яті не перевищує 1Gb;
- об’єм дискового простору не перевищує 25Gb;
- кількість ядер центрального процесору не більше 1.

Подібне середовище відповідає мінімальній конфігурації дроплету Digital Ocean. Отже, якщо програмне забезпечення коректно функціонує у такому середовищі, то його можна розгорнути для промислового використання з невеликими щомісячними економічними витратами(станом на 17.05.2020 5 доларів США на місяць, відповідно до політики цін платформи DigitalOcean).

3.3 Функціональність, що підлягає тестуванню

Функції ПЗ, що підлягають тестуванню:

- реєстрація у застосунку;
- авторизація у застосунку;
- демонстрація переліку тегів;
- демонстрація переліку метрик;
- демонстрація переліку категорій;
- демонстрація переліку подій;
- маніпулювання тегами;
- маніпулювання метриками;
- маніпулювання категоріями;
- маніпулювання подіями;
- демонстрація кругової діаграми, що показує частку подій за категоріями;
- демонстрація кругової діаграми, що показує частку подій за тегами;
- демонстрація лінійної діаграми, що показує значення метрик у часі.

3.4 Функціональність, що не підлягає тестуванню

Не підлягає тестуванню така функціональність:

- accessibility (можливість використання застосунку людьми з обмеженими можливостями);
- learnability (властивість застосунку, за наявності якої, користувачам легко ознайомитися з застосунком та навчитися добре використовувати всі його функції та можливості).

3.5 Методологія проведення тестування

Тестування мусить бути проведено для кожного прецеденту (test case). Інформація про перебіг та результат кожного тесту буде збережено до сервісу qase.io. Тестувальник проводить тест та позначає результат тестування за кожним прецедентом.

Після завершення тестування, проводиться перегляд звіту тестування. У разі виявлення помилок – перелік помилок передається розробнику, для виправлення.

3.6 Критерії проходження/провалу тестування

Вся основна функціональність системи повинна функціонувати як очікувалося та окреслено в окремих тестових випадках. Не повинно бути виявлених критичних дефектів. Кінцевий користувач повинен мати змогу успішно завершити цикл реєстрації, авторизації, створення структуруючих сутностей та внесення інформації про події. 90% усіх тестових випадків повинні бути успішно пройдені. Жоден невдалий випадок не повинен мати вирішальне значення для можливості кінцевого користувача використовувати додаток.

3.7 Критерії призупинення тестування

Процес тестування невідкладно припиняють, якщо у системі спостерігаються дефекти у авторизації або реєстрації користувача.

3.8 Вимоги до тестового середовища

БД застосунку повинна містити мінімально необхідний для проведення тестування набір сутностей(включаючи, але не обмежуючись, тестовим користувачем).

3.9 Відповідальність

Відповідальним за дотримання розкладу тестування, своєчасне виявлення дефектів у програмному забезпеченні є розробник застосунку.

3.10 Розклад

Процес тестування розпочинається за 2 тижні до дати здачі програмного продукту в експлуатацію.

3.11 Ризики та непередбачувані ситуації

Тестування може бути відкладено або не проведено у випадку дії непереборної сили або форс-мажорних обставин. Перелік обставин непереборної сили (форс-мажору) міститься в п. 3.1. Регламенту засвідчення Торгово-промисловою палатою України та регіональними торгово-промисловими палатами форс-мажорних обставин (обставин непереборної сили), зі змінами та доповненнями від 26.04.2016 р. [12]. У такому випадку розробник застосунку не несе відповідальності за несвоєчасне проведення тестування.

3.12 Схвалення

Право констатувати успішне завершення процесу тестування програмного забезпечення та визначати момент, коли розроблене програмне забезпечення набуває властивостей, що роблять його придатним до промислової експлуатації, залишається за розробником програмного забезпечення.

3.13 Опис тестових прецедентів

В таблицях нижче наведено перелік основних тестових прецедентів, для яких проводиться тестування. Назва прецеденту відповідає його назві у системі зберігання тестових прецедентів, що використовувалась.

Таблиця 3.1 – Тестовий прецедент “Авторизація”

Назва	Авторизація
Передумови	- Користувач неавторизований у системі. - Тестувальник знає валідний логін та пароль.
Кроки виконання	- Відкрити бокове меню застосунку. - Натиснути кнопку “Sign in”. - Заповнити необхідні поля вводу. - Натиснути кнопку “Sign in”.
Очікуваний результат	Демонструється повідомлення про успішну авторизацію. Користувач перенаправляється на головну сторінку додатку.
Отриманий результат	Демонструється повідомлення про успішну авторизацію. Користувач перенаправляється на головну сторінку додатку.

Таблиця 3.2 – Тестовий прецедент “Реєстрація”

Назва	Реєстрація
Передумови	Користувач неавторизований у системі.
Кроки виконання	- Відкрити бокове меню застосунку. - Натиснути кнопку “Sign up”. - Заповнити необхідні поля вводу. - Натиснути кнопку “Sign up”.
Очікуваний результат	Демонструється повідомлення про успішну реєстрацію. Користувач перенаправляється на сторінку авторизації.
Отриманий результат	Демонструється повідомлення про успішну реєстрацію. Користувач перенаправляється на сторінку авторизації.

Таблиця 3.3 – Тестовий прецедент “Створення тегу”

Назва	Створення тегу
Передумови	Користувач авторизований у системі.
Кроки виконання	- Відкрити бокове меню застосунку. - Натиснути кнопку “Tags”. - Натиснути кнопку “New”. - Заповнити необхідні поля вводу у діалоговому вікні. - Натиснути кнопку “Create”.

Продовження таблиці 3.3

Очікуваний результат	Демонструється повідомлення про успішне створення тегу. Тег демонструється у переліку тегів користувача.
Отриманий результат	Демонструється повідомлення про успішне створення тегу. Тег демонструється у переліку тегів користувача.

Таблиця 3.4 – Тестовий прецедент “Створення тегу”

Назва	Створення метрики
Передумови	Користувач авторизований у системі.
Кроки виконання	- Відкрити бокове меню застосунку. - Натиснути кнопку “Metrics”. - Натиснути кнопку “New”. - Заповнити необхідні поля вводу у діалоговому вікні. - Натиснути кнопку “Create”.
Очікуваний результат	Демонструється повідомлення про успішне створення метрики. Метрика демонструється у списку метрик користувача.
Отриманий результат	Демонструється повідомлення про успішне створення метрики. Метрика демонструється у списку метрик користувача.

Таблиця 3.5 – Тестовий прецедент “Створення метрики”

Назва	Створення метрики
Передумови	Користувач авторизований у системі.
Кроки виконання	- Відкрити бокове меню застосунку. - Натиснути кнопку “Metrics”. - Натиснути кнопку “New”. - Заповнити необхідні поля вводу у діалоговому вікні. - Натиснути кнопку “Create”.
Очікуваний результат	Демонструється повідомлення про успішне створення метрики. Метрика демонструється у списку метрик користувача.
Отриманий результат	Демонструється повідомлення про успішне створення метрики. Метрика демонструється у списку метрик користувача.

Таблиця 3.6 – Тестовий прецедент “Створення категорії”

Назва	Створення категорії
Передумови	Користувач авторизований у системі.
Кроки виконання	- Відкрити бокове меню застосунку. - Натиснути кнопку “Categories”. - Натиснути кнопку “New”. - Заповнити необхідні поля вводу у діалоговому вікні. - Натиснути кнопку “Create”.

Продовження таблиці 3.6

Очікуваний результат	Демонструється повідомлення про успішне створення категорії. Категорія демонструється у списку категорій користувача.
Отриманий результат	Демонструється повідомлення про успішне створення категорії. Категорія демонструється у списку категорій користувача.

Таблиця 3.7 – Тестовий прецедент “Створення події”

Назва	Створення метрики
Передумови	Користувач авторизований у системі.
Кроки виконання	- Відкрити бокове меню застосунку. - Натиснути кнопку “Events”. - Натиснути кнопку “New”. - Заповнити необхідні поля вводу у діалоговому вікні. - Натиснути кнопку “Create”.
Очікуваний результат	Демонструється повідомлення про успішне створення події. Подія демонструється у списку подій користувача.
Отриманий результат	Демонструється повідомлення про успішне створення події. Подія демонструється у списку подій користувача.

Таблиця 3.8 – Тестовий прецедент “Перегляд кругової діаграми за категоріями”

Назва	Перегляд кругової діаграми за категоріями
Передумови	Користувач авторизований у системі.
Кроки виконання	- Відкрити бокове меню застосунку. - Натиснути кнопку “Statistics”. - Натиснути кнопку “By categories”. - Обрати необхідний діапазон дат.
Очікуваний результат	Демонстрація кругової діаграми, що показує частку подій за кожною категорією користувача.
Отриманий результат	Демонстрація кругової діаграми, що показує частку подій за кожною категорією користувача.

Таблиця 3.9 – Тестовий прецедент “Перегляд кругової діаграми за тегами”

Назва	Перегляд кругової діаграми за тегами
Передумови	Користувач авторизований у системі.
Кроки виконання	- Відкрити бокове меню застосунку. - Натиснути кнопку “Statistics”. - Натиснути кнопку “By tags”. - Обрати необхідний діапазон дат.

Продовження таблиці 3.9

Очікуваний результат	Демонструється кругова діаграма, що показує частку подій за кожним тегом користувача.
Отриманий результат	Демонструється кругова діаграма, що показує частку подій за кожним тегом користувача.

Таблиця 3.10 – Тестовий прецедент “Перегляд графіку зміни значень метрик у часі”

Назва	Перегляд графіку зміни значень метрик у часі
Передумови	Користувач авторизований у системі.
Кроки виконання	- Відкрити бокове меню застосунку. - Натиснути кнопку “Statistics”. - Натиснути кнопку “Metrics monitoring”. - Для кожної метрики, динаміку змін якої необхідно показати на графіку: 1) увімкнути перемикач “Display metric”; 2) обрати агрегуючу функцію з випадаючого списку; 3) обрати нормалізуюче перетворення з випадаючого списку. - Натиснути кнопку “Apply”.

Продовження таблиці 3.10

Очікуваний результат	Демонструється графік, що показує динаміку змін значень обраних метрик у часі з кроком в один місяць.
Отриманий результат	Демонструється графік, що показує динаміку змін значень обраних метрик у часі з кроком в один місяць.

3.14 Висновки по розділу

У даному розділі було проведено планування тестування програмного забезпечення Biograph, що дозволить виявити всі потенційні дефекти, які можуть призвести до небажаного відтоку користувачів або погіршення їх досвіду використання застосунку.

В результаті проведеного (відповідно до тест плану) тестування було з’ясовано, що робота застосунку відповідає очікуваній та описаній у вимогах поведінці. Отже, розроблений застосунок придатний до промислової експлуатації.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Оскільки веб-застосунок складається з двох частин – клієнтської та серверної, доцільно розглянути процес впровадження кожної частини окремо.

4.1.1 Розгортання серверної частини

Серверна частина додатку потребує доступу до СКБД PostgreSQL. Рекомендована версія - PostgreSQL 12.2-2. Серверна частина програмного продукту надається у форматі виконуваного JAR-файлу. Такий формат дозволяє запустити серверну частину застосунку на сервері, що має містить середовище JRE. Найпростіший процес запуску передбачає запуск виконуваного jar-файлу за допомогою інструментів, що надаються середовищем JRE (зазвичай, для цього використовують утиліту jar). Однак, рекомендований спосіб розгортання – використання хмарних сервісів, що використовують модель хмарних обчислень PAAS. Рекомендується використати для цього платформу Heroku.

4.1.2 Розгортання клієнтської частини

Клієнтська частина застосунку надається у вигляді необхідних html-, css-, та js-файлів, що дозволяє використовувати веб-сервери для забезпечення доступу користувачів до клієнтської частини. Розробник програмного забезпечення рекомендує використовувати хмарні сервіси, що використовують модель хмарних обчислень PAAS. Для вирішення задачі розгортання та введення у промислову експлуатацію клієнтської частини застосунку можна використати платформу Firebase. Можливості Firebase

Hosting цілком задовольняють потреби, що виникають при розгортанні клієнтської частини застосунку Biograph.

4.1.3 Забезпечення захищеного каналу зв’язку

У будь-якому випадку, незалежно від обраного способу розгортання, необхідно обов’язково забезпечити захищений канал зв’язку між клієнтською та серверною частинами застосунку, а також сервером, що надає клієнтську частину та користувачем системи. Для цього рекомендується використовувати протокол HTTPS, сформовавши криптографічні ключі та SSL-сертифікати для серверної частини додатку. Розробник програмного забезпечення рекомендує використовувати безкоштовні сертифікати, що надаються за вимогою акредитованим центром сертифікації ключів Let’s Encrypt.

Особливої уваги потребує з’єднання між СКБД та серверною частиною застосунку. У випадку, якщо серверна частина застосунку та СКБД розгорнуті на різних комп’ютерах, необхідно забезпечити захищений канал зв’язку між ними, способом, аналогічним до того, що описаний вище.

4.2 Робота з програмним забезпеченням

Детальна інструкція з експлуатації застосунку наведена у документі КПІ.ІП-6326.045420.06.34 «Керівництво користувача».

ВИСНОВКИ

В рамках даної роботи було розглянуто підходи до покращення якості життя використовуючи програмне забезпечення. Було проведено аналіз предметної області, знайдено та проаналізовано відомі підходи до вирішення задачі керування якістю, структурування інформації та її аналізу. Було розглянуто відомі індустріальні рішення до побудови подібного програмного забезпечення, проаналізовано переваги та знайдено недоліки відомих програмних продуктів, що придатні для вирішення проблеми керування якістю життя людини.

Після детального вивчення предметної області, відомих індустріальних рішень та програмних продуктів було проведено аналіз вимог до програмного забезпечення. Проведений аналіз, вже наявних, подібних за призначенням програмних продуктів, дав можливість зрозуміти їх недоліки, та спроектувати програмний продукт, що не має зовсім, або нівелює значущість таких недоліків. Таким чином, розробка такого програмного продукту може бути економічно вигідною, оскільки цільова аудиторія наявних програмних продуктів може зацікавитись продуктом, що вбираючи в собі переваги існуючої програми, позбавляє від незручностей її використання, оскільки має меншу кількість та меншу значущість недоліків.

Спроектований програмний продукт має такі переваги, перед вже існуючими:

- зручність використання як на мобільних так і на стаціонарних пристроїв;
- гнучкість структурування інформації, орієнтована на потреби конкретного користувача;
- більші, порівняно з потенційними конкурентами, можливості візуалізації інформації;
- можливість доступу до даних онлайн.

Після формулювання вимог до програмного забезпечення, було проведено проектування архітектури програмного забезпечення, що дозволить не тільки задовольнити сформульовані вимоги, але і передбачає можливість подальшого розширення функціональності продукту. Наступним етапом було проведено розробку програмного продукту, відповідно до спроектованої архітектури. В процесі розробки було використано підхід безперервної доставки(Continious Delivery), що дозволяє швидко розгортати поточну версію програмного забезпечення у середовищі, що за своїми характеристиками близьке до реального експлуатаційного середовища. В даній роботі реалізація такого підходу забезпечувалась хмарними платформами Heroku та Firebase Hosting. Цей підхід було використано, щоб отримати можливість швидко та легко тестувати розроблену функціональність у середовищі, що наближене до реального.

Після завершення розробки було проведено тестування створеного продукту, відповідно до розробленого плану тестування. Метою такого тестування була необхідність впевнитись, що розроблений продукт повністю відповідає сформульованим вимогам і може бути переданий до промислової експлуатації. Розроблене програмне забезпечення успішно пройшло тестування, отже може вважатись таким, що виконує покладені на нього задачі.

Тому, мету даного проекту вважаю досягнутою.

ПЕРЕЛІК ПОСИЛАНЬ

1) Цикл Шухарта — Демінг [Електронний ресурс] // Wikipedia. – 2020. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%A6%D0%B8%D0%BA%D0%BB_%D0%A8%D1%83%D1%85%D0%B0%D1%80%D1%82%D0%B0_%E2%80%94_%D0%94%D0%B5%D0%BC%D1%96%D0%BD%D0%B3%D0%B0.

2) Бази даних і бази знань у державному управлінні – Київ: НАДУ, 2007. – 104 с.

3) Інформаційна система [Електронний ресурс] // Wikipedia. – 2020. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0.

4) Інженерія знань [Електронний ресурс] // Wikipedia. – 2020. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D1%96%D1%8F_%D0%B7%D0%BD%D0%B0%D0%BD%D1%8C.

5) База знань [Електронний ресурс] // Wikipedia. – 2020. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B7%D0%BD%D0%B0%D0%BD%D1%8C.

6) Онтологія (інформатика) [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9E%D0%BD%D1%82%D0%BE%D0%BB%D0%BE%D0%B3%D1%96%D1%8F_(%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0).

7) Мапирование реляционной модели в метамодель [Електронний ресурс] // Studbooks – Режим доступу до ресурсу:

https://studbooks.net/1998567/informatika/mapirovanie_relyatsionnoy_modeli_metamodel#43.

8) Унаочнювання даних [Електронний ресурс] // Wikipedia. – 2020. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%A3%D0%BD%D0%B0%D0%BE%D1%87%D0%BD%D1%8E%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85.

9) Реактивне програмування [Електронний ресурс] // Wikipedia. – 2020. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B0%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F.

10) Модель-вид-контролер [Електронний ресурс] // Wikipedia. – 2020. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C-%D0%B2%D0%B8%D0%B4-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5%D1%80.

11) State management [Електронний ресурс] // Wikipedia. – 2020. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/State_management.

12) Регламент засвідчення Торгово-промисловою палатою України та регіональними торгово-промисловими палатами форс-мажорних обставин (обставин непереборної сили) [Електронний ресурс] // ПРЕЗИДІЯ ТОРГОВО-ПРОМИСЛОВОЇ ПАЛАТИ УКРАЇНИ. – 2014. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0040571-14.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

ВЕБ-ЗАСТОСУВАННЯ МОНІТОРИНГУ ЖИТТЄВИХ ПОДІЙ

Технічне завдання

КПІ.ІП-6326.045420.03.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____О.В.Ковтунець

Виконавець:

_____А.О. Семченко

Нормоконтроль:

_____ К.І. Ліщук

Київ – 2020 року

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Веб застосування моніторингу життєвих подій

Галузь застосування:

Наведене технічне завдання поширюється на розробку веб застосування “Biograph”, яке використовується для зберігання, моніторингу та аналізу життєвого досвіду користувача. Користувачами можуть бути люди, що активно планують та відстежують своє життя. Застосування також може бути використано для проведення ретроспекції життєвих подій та виявлення тенденцій та закономірностей.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки “Biograph” є завдання на дипломне проектування, затверджене кафедрою автоматизованих систем обробки інформації і управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» («КПІ ім. Ігоря Сікорського»).

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для аналізу та структурованого зберігання інформації про життєві події. Отримані дані можуть бути використані користувачем для покращення якості свого життя.

Метою проведення розробки є надання користувачу застосунку можливості збереження життєвого досвіду та його аналізу, з можливістю виконувати вищезазначені дії онлайн зі стаціонарних та мобільних пристроїв.

Для виконування описаних вище цілей програмне забезпечення повинно виконувати наступні функції:

- можливість структурованого збереження інформації про подію;
- пошук інформації про подію;
- представлення інформації у зручному для аналізу вигляді.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

4.1.1 Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

- реєстрація у системі;
- вхід у систему;
- визначення структури подій;
- збереження інформації про подію;
- пошук подій;
- відображення статистики.

4.1.2 Вимоги до організації вхідних даних

Вхідні дані представлені наступними елементами

- перелік метрик (ознак) подій;
- перелік категорій подій (з вказанням метрик);
- перелік можливих тегів;
- перелік подій.

4.1.3 Розробку виконати на платформі JVM

4.1.4 Додаткові вимоги не висуваються

4.2 Вимоги до надійності

4.2.1 Передбачити контроль введення інформації.

4.2.2 Передбачити захист від некоректних дій користувача.

4.3 Умови експлуатації

4.3.1 Умови експлуатації згідно СанПін 2.2.2.542 – 96.

Не висуваються.

4.3.2 Обслуговування та обслуговуючий персонал.

Не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

4.4.1 Програмне забезпечення повинно функціонувати на серверах з операційною системою Ubuntu. Клієнтська частина повинна функціонувати в браузерах:

– Google Chrome версії 81.0;

– Mozilla Firefox версії 76.0.1.

4.4.2 Мінімальна конфігурація технічних засобів:

4.4.2.1 Тип процесору

Будь-який процесор що підтримується ОС Ubuntu та JVM 8.

Мінімальна підтримувана кількість ядер - 1

4.4.2.2 Об‘єм ОЗП

1024 Мб, або більше.

4.5 Вимоги до інформаційної та програмної сумісності

– Серверне програмне забезпечення повинно працювати під управлінням операційної системи Ubuntu.

– Клієнтське програмне забезпечення повинно працювати коректно:

а) у браузерах:

– Google Chrome версії 81.0;

– Mozilla Firefox версії 76.0.1.

б) на пристроях з розподільною здатністю екрану:

– 1920 * 1080 пікселів;

– 1366 * 798 пікселів.

– Вхідні дані повинні бути представлені в наступному форматі:

1) інформація про подію представлена:

а) назвою події;

б) текстовим описом події;

в) інтервалом часу, протягом якого відбувалась подія;

г) значеннями метрик події;

д) переліком тегів події.

2) інформація про метрику події представлена:

а) назвою метрики;

б) текстовим описом метрики;

в) типом метрики (числовий або номінальний);

г) обмеженнями значення метрики (якщо необхідно).

3) Інформація про тег представлена:

а) назвою тегу;

б) текстовим описом тегу;

в) кольором тегу.

4) Інформація про категорію події представлена:

а) назвою категорії;

б) текстовим описом категорії;

в) переліком метрик категорії;

г) кольором категорії.

– Доступ до серверу повинен здійснюватися тільки через веб-інтерфейс.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Згенерувати установочну версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

а) Супроводжувальна документація:

1) пояснювальна записка;

2) керівництво користувача;

3) програма та методика тестування.

б) Довідникова документація:

1) програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі;

в) Графічна документація:

1) схема структурна станів системи;

2) схема структурна бази даних;

3) схема структурна класів програмного забезпечення;

4) креслення елементів графічного інтерфейсу системи.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	10.02.2020	
2.	Розробка технічного завдання	24.02.2020	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	05.03.2020	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	19.03.2020	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	02.04.2020	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	16.04.2020	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	30.04.2020	Пояснювальна записка.
8.	Розробка матеріалів графічної частини проекту	14.05.2020	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	27.05.2020	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

7.1 Види випробувань

Тестування описано в пояснювальній записці «КПІ.ІП-6326.045420.02.81», розділі 3 «Аналіз та тестування програмного забезпечення».

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

ВЕБ-ЗАСТОСУВАННЯ МОНІТОРИНГУ ЖИТТЄВИХ ПОДІЙ

Опис програми

КПІ.ІП-6326.045420.04.13

“ПОГОДЖЕНО”

Керівник проєкту:

_____ О.В. Ковтунець

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____А.О. Семченко

Київ – 2020 року

Тексти програмного коду

Веб-застосування моніторингу життєвих подій

(Найменування програми (документа))

DVD-R

(Вид носія даних)

51 арк, 250 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2020

```
package fun.asem.biograph.webapp.controller;

import fun.asem.biograph.webapp.domain.Attribute;
import fun.asem.biograph.webapp.domain.User;
import fun.asem.biograph.webapp.dto.mapper.AttributeMapper;
import fun.asem.biograph.webapp.dto.model.attribute.CreateAttributeDto;
import fun.asem.biograph.webapp.dto.model.attribute.ResponseAttributeDto;
import fun.asem.biograph.webapp.service.attribute.AttributeService;
import fun.asem.biograph.webapp.service.user.UserService;
import lombok.RequiredArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;
import java.security.Principal;
import java.util.List;
import java.util.stream.Collectors;

@RequestMapping("/api/")
@RestController
@RequiredArgsConstructor
public class AttributeController extends BaseController {
    private final UserService userService;
    private final AttributeService attributeService;
    private final AttributeMapper dtoMapper;

    /**
     * @return Returns list of all attributes that are owned by current user
     */
    @GetMapping("/users/{userId}/attributes")
    public List<ResponseAttributeDto> getUserAttributes(@PathVariable Long userId, Principal principal) {
        User user = getUser(principal);
        checkAccess(userId, user);
        return attributeService.getAllAttributesOwnedByUser(user)
                .stream()
                .map(this::convertToDto)
                .collect(Collectors.toList());
    }

    @ResponseStatus(HttpStatus.CREATED)
```

```
    @PostMapping("/users/{userId}/attributes")
    public ResponseAttributeDto createAttribute(
            @RequestBody @Valid CreateAttributeDto createAttributeDto,
            @PathVariable Long userId,
            Principal principal) {
        Attribute attribute = convertToEntity(createAttributeDto);
        User user = getUser(principal);
        checkAccess(userId, user);
        return convertToDto(attributeService.create(attribute, user));
    }

    private User getUser(Principal principal) {
        return userService.getUserByUserDetails(principal.getName());
    }

    private Attribute convertToEntity(CreateAttributeDto dto) {
        return dtoMapper.createDtoToAttribute(dto);
    }

    private ResponseAttributeDto convertToDto(Attribute attribute) {
        return dtoMapper.attributeToDto(attribute);
    }
}
package fun.asem.biograph.webapp.controller;

import fun.asem.biograph.webapp.dto.model.AuthorizationRequest;
import fun.asem.biograph.webapp.dto.model.ErrorDescription;
import fun.asem.biograph.webapp.dto.model.RegistrationRequest;
import fun.asem.biograph.webapp.dto.model.ServerResponse;
import fun.asem.biograph.webapp.service.registration.AuthService;
import fun.asem.biograph.webapp.service.user.UserService;
import fun.asem.biograph.webapp.util.security.jwt.JwtTokenUtil;
import lombok.RequiredArgsConstructor;
import org.springframework.security.authentication.*;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.PostMapping;
```

```
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import javax.validation.Valid;
import java.util.Objects;

@RequiredArgsConstructor
@RequestMapping("/api/auth")
@RestController
@Validated
public class AuthController {
    private final AuthenticationManager authenticationManager;
    private final AuthService authService;
    private final JwtTokenUtil jwtTokenUtil;
    private final UserDetailsService userDetailsService;
    private final UserService userService;

    @PostMapping("/signUp")
    public ServerResponse signUp(@RequestBody @Valid RegistrationRequest registrationRequest) {
        return authService.signUp(registrationRequest);
    }

    @PostMapping("/signIn")
    public ServerResponse signIn(@RequestBody @Valid AuthorizationRequest authorizationRequest) {
        ServerResponse response;
        try {
            // checking user credentials
            authenticateWithSpringAuthenticationManager(authorizationRequest.getEmail(),
authorizationRequest.getPassword());
            // getting user info ( needed for token issuing process )
            UserDetails userDetails = userDetailsService.loadUserByUsername(authorizationRequest.getEmail());
            // issue token to client and send it
            String token = jwtTokenUtil.generateToken(userDetails);
            response = ServerResponse.builder()
                    .status(ServerResponse.ResponseStatus.OK)
                    .build();
            response.setData(userService.getUserByUserDetails(userDetails.getUsername()));
            response.setAuthToken("Bearer " + token);
```

```
        } catch (DisabledException e) {
            response = ServerResponse.builder()
                    .status(ServerResponse.ResponseStatus.ERROR)
                    .build();
            response.setData(ErrorDescription.builder().message("Your account was disabled because of an suspicious activity").build());
        } catch (BadCredentialsException e) {
            response = ServerResponse.builder()
                    .status(ServerResponse.ResponseStatus.ERROR)
                    .build();
            response.setData(ErrorDescription.builder().message("Invalid credentials").build());
        } catch (CredentialsExpiredException e) {
            response = ServerResponse.builder()
                    .status(ServerResponse.ResponseStatus.ERROR)
                    .build();
            response.setData(ErrorDescription.builder().message("Credentials has been expired").build());
        } catch (AuthenticationException e) {
            response = ServerResponse.builder()
                    .status(ServerResponse.ResponseStatus.ERROR)
                    .build();
            response.setData(ErrorDescription.builder().message("Authentication error").build());

        }
        return response;
    }

    private void authenticateWithSpringAuthenticationManager(String email, String password) {
        Objects.requireNonNull(email, "Email value is expected");
        Objects.requireNonNull(password, "Password value is expected");
        // throws an exception if credentials are invalid
        authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(email, password));
    }
}
package fun.asem.biograph.webapp.controller;

import fun.asem.biograph.webapp.domain.User;
import fun.asem.biograph.webapp.exception.UnauthorizedException;

public abstract class BaseController {
```

```
    protected void checkAccess(Long userId, User user) {
        if (!user.getUserId().equals(userId)) {
            throw new UnauthorizedException();
        }
    }
}
package fun.asem.biograph.webapp.controller;

import fun.asem.biograph.webapp.domain.User;
import fun.asem.biograph.webapp.dto.model.category.CreateCategoryDto;
import fun.asem.biograph.webapp.dto.model.category.ResponseCategoryDto;
import fun.asem.biograph.webapp.service.category.CategoryService;
import fun.asem.biograph.webapp.service.user.UserService;
import lombok.RequiredArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;
import java.security.Principal;
import java.util.List;

@RequiredArgsConstructor
@RestController
@RequestMapping("/api")
@Validated
public class CategoryController extends BaseController {
    private final CategoryService categoryService;
    private final UserService userService;

    @ResponseStatus(HttpStatus.CREATED)
    @PostMapping("/users/{userId}/categories")
    public ResponseCategoryDto create(
            @RequestBody @Valid CreateCategoryDto categoryDto,
            @PathVariable Long userId,
            Principal principal) {
        User currentUser = userService.getUserByUserDetails(principal.getName());
        checkAccess(userId, currentUser);
        return categoryService.create(categoryDto, currentUser);
```

```
    }

    @GetMapping("/users/{userId}/categories")
    public List<ResponseCategoryDto> getUserCategories(@PathVariable Long userId, Principal principal) {
        User currentUser = userService.getUserByUserDetails(principal.getName());
        checkAccess(userId, currentUser);
        return categoryService.getUserCategories(currentUser);
    }
}
package fun.asem.biograph.webapp.controller;

import fun.asem.biograph.webapp.domain.User;
import fun.asem.biograph.webapp.dto.model.event.CreateEventDto;
import fun.asem.biograph.webapp.dto.model.event.ResponseEventDto;
import fun.asem.biograph.webapp.service.event.EventService;
import fun.asem.biograph.webapp.service.user.UserService;
import lombok.RequiredArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;
import java.security.Principal;
import java.util.List;

@RequiredArgsConstructor
@RestController
@RequestMapping("/api")
@Validated
public class EventController extends BaseController {
    private final EventService eventService;
    private final UserService userService;

    @ResponseStatus(HttpStatus.CREATED)
    @PostMapping("/users/{userId}/events")
    public ResponseEventDto create(
            @RequestBody @Valid CreateEventDto eventDto,
            @PathVariable Long userId,
            Principal principal) {
```

```
        User currentUser = userService.getUserByUserDetails(principal.getName());
        checkAccess(userId, currentUser);
        return eventService.create(eventDto, currentUser);
    }

    @GetMapping("/users/{userId}/events")
    public List<ResponseEventDto> getUserEvents(@PathVariable Long userId, Principal principal) {
        User currentUser = userService.getUserByUserDetails(principal.getName());
        checkAccess(userId, currentUser);
        return eventService.getUserEvents(currentUser);
    }
}
package fun.asem.biograph.webapp.controller;

import fun.asem.biograph.webapp.service.storage.google.GoogleStorageProviderService;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;

import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServletRequest;
import java.util.NoSuchElementException;

@RequiredArgsConstructor
@RequestMapping("/oauth2")
@Controller
public class OAuthController {
    private final GoogleStorageProviderService googleService;

    @GetMapping("/google")
    public String acceptCode(HttpServletRequest request) {
        String code = request.getParameter("code");
        googleService.acceptOauth2CallbackCode(code, getUserId(request).toString());
        return "provider-added";
    }

    private Long getUserId(HttpServletRequest request) {
        for (Cookie cookie : request.getCookies()) {
```

```
            // FIXME asem REFACTOR - get rid of this magic constant - "userId"
            if (cookie.getName().equals("userId")) {
                return Long.parseLong(cookie.getValue());
            }
        }
        throw new NoSuchElementException("Cookie value 'userId' should be present");
    }
}
package fun.asem.biograph.webapp.controller;

import com.google.api.services.drive.model.File;
import com.google.api.services.drive.model.FileList;
import fun.asem.biograph.webapp.domain.User;
import fun.asem.biograph.webapp.service.storage.StorageProviderService;
import fun.asem.biograph.webapp.service.user.UserService;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;

import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.List;
import java.util.NoSuchElementException;
import java.util.Optional;

@Slf4j
@RequiredArgsConstructor
@RequestMapping("/api/storage")
@Controller
public class StorageProviderController {
    private final StorageProviderService storageProviderService;
```

```
    private final UserService userService;

    @PostMapping("/connect")
    public ResponseEntity<Object> connectStorageProvider(@RequestParam(name = "provider", required =
true) String providerName,
                                                         @RequestParam(name = "userId", required = true) Long userId) {
        if (providerName.equalsIgnoreCase("google")) {
            Optional<User> user = userService.findUserByUserId(userId);
            String                                                redirectUrl                                                =
user.map(storageProviderService::getAuthorizationUrl).orElseThrow(NoSuchElementException::new);
            // preparing redirect response
            HttpHeaders headers = new HttpHeaders();
            headers.add("Location", redirectUrl);
            // saving userId to cookie with lifetime = 60 seconds
            // FIXME asem SECURITY - change Max-Age back to 60 seconds
            headers.add("Set-Cookie", "userId=" + userId.toString() + ";Max-Age=3600; Path=/");
            return new ResponseEntity<>(headers, HttpStatus.TEMPORARY_REDIRECT);
        } else {
            throw new UnsupportedOperationException("Unknown provider: " + providerName);
        }
    }

    @PostMapping("uploadFile")
    public String uploadFile(@RequestPart("file") MultipartFile file, HttpServletRequest request) throws
IOException {
        log.info("Got upload file request from user with id={}", getUserId(request));
        storageProviderService.uploadFile(file.getName(), file.getInputStream(), getUserId(request).toString());
        return "redirect:/api/storage/listFiles";
    }

    /* FIXME asem REFACTOR duplicate method - the same in OAuthController */
    private Long getUserId(HttpServletRequest request) {
        for (Cookie cookie : request.getCookies()) {
            // FIXME asem REFACTOR - get rid of this magic constant - "userId"
            if (cookie.getName().equals("userId")) {
                return Long.parseLong(cookie.getValue());
            }
        }
        throw new NoSuchElementException("Cookie value 'userId' should be present");
```

```
    }

    @GetMapping("listFiles")
    public String listFiles(Model model, HttpServletRequest request) {
        FileList files = storageProviderService.listFiles(getUserId(request).toString());
        List<File> filesFiles = files.getFiles();
        File file = filesFiles.get(0);
        model.addAttribute("files", files.getFiles());
        return "listFiles";
    }

    @GetMapping("downloadFile")
    public void downloadFile(@RequestParam(name = "fileId") String fileId, HttpServletRequest request,
HttpServletResponse response) throws IOException {
        storageProviderService.downloadFileInto(fileId, getUserId(request).toString(),
response.getOutputStream());
        response.flushBuffer();
    }
}
package fun.asem.biograph.webapp.controller;

import fun.asem.biograph.webapp.domain.User;
import fun.asem.biograph.webapp.dto.model.tag.CreateTagDto;
import fun.asem.biograph.webapp.dto.model.tag.ResponseTagDto;
import fun.asem.biograph.webapp.service.tag.TagService;
import fun.asem.biograph.webapp.service.user.UserService;
import lombok.RequiredArgsConstructor;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;
import java.security.Principal;
import java.util.List;

@RequiredArgsConstructor
@RestController
@RequestMapping("/api")
@Validated
public class TagController extends BaseController {
```

```
    private final UserService userService;
    private final TagService tagService;

    @GetMapping("/users/{userId}/tags")
    public List<ResponseTagDto> getUserTags(@PathVariable Long userId, Principal principal) {
        User currentUser = userService.getUserByUserDetails(principal.getName());
        checkAccess(userId, currentUser);
        return tagService.getUserTags(currentUser);
    }

    @PostMapping("/users/{userId}/tags")
    public ResponseTagDto create(@RequestBody @Valid CreateTagDto createTagDto, @PathVariable Long
userId, Principal principal) {
        User currentUser = userService.getUserByUserDetails(principal.getName());
        checkAccess(userId, currentUser);
        return tagService.createTag(createTagDto, currentUser);
    }
}
package fun.asem.biograph.webapp.controller;

import fun.asem.biograph.webapp.dto.model.ServerResponse;
import org.springframework.security.core.Authentication;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@RequestMapping("/home")
@Controller
public class TestController {
    @GetMapping("")
    public String test(Authentication authentication) {
        return "home";
    }

    @ResponseBody
    @GetMapping("/secured")
    public ServerResponse securedTest() {
        return ServerResponse
```

```
                .builder()
                .status(ServerResponse.ResponseStatus.OK)
                .data("You are now authenticated")
                .build();
    }
}
package fun.asem.biograph.webapp.configuration;

import fun.asem.biograph.webapp.dto.model.AuthorizationRequest;
import fun.asem.biograph.webapp.security.jwt.request_filter.AuthenticationFilter;
import lombok.RequiredArgsConstructor;
import org.springframework.context.annotation.Bean;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.AuthenticationEntryPoint;
import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

@RequiredArgsConstructor
@EnableWebSecurity
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {
    private final UserDetailsService userDetailsService;
    private final AuthenticationEntryPoint authenticationEntryPoint;
    private final AuthenticationFilter authenticationRequestFilter;

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        // FIXME asem find or implement custom password encoder that supports SHA3+Argon+AES128
```

```
        return new BCryptPasswordEncoder();
    }

    /**
     * Just to provide AuthenticationManager bean to application context
     *
     * @see fun.asem.biograph.webapp.controller.AuthController#signIn(AuthorizationRequest)
     */
    @Bean
    @Override
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.authenticationManagerBean();
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
                .cors()
                .and()
                .csrf().disable()
                .authorizeRequests()
                // TODO asem probably "/home" endpoint should be removed from here in production - it is here just
for test
                .antMatchers(
                        "/api/auth/**",
                        "/oauth2/**",
                        "/api/storage/connect",
                        "/api/storage/uploadFile",
                        "/api/storage/listFiles",
                        "/api/storage/downloadFile",
                        "/home/"
                ).permitAll()
                .anyRequest().authenticated()
                .and()
                .exceptionHandling().authenticationEntryPoint(authenticationEntryPoint)
                .and()
                .sessionManagement()
                .sessionCreationPolicy(SessionCreationPolicy.STATELESS);
        http.addFilterBefore(authenticationRequestFilter, UsernamePasswordAuthenticationFilter.class);
```

```
    }
}
package fun.asem.biograph.webapp.configuration;

import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class WebConfiguration implements WebMvcConfigurer {
    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/**")
                .allowedMethods("*");
    }
}
package fun.asem.biograph.webapp.repository;

import fun.asem.biograph.webapp.domain.Attribute;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;
import java.util.Optional;

public interface AttributeRepository extends JpaRepository<Attribute, Long> {
    List<Attribute> findAllByAttributeIdIn(List<Long> attributeIds);

    Optional<Attribute> findByAttributeId(Long attributeId);
}
package fun.asem.biograph.webapp.repository;

import fun.asem.biograph.webapp.domain.Category;
import org.springframework.data.jpa.repository.JpaRepository;

public interface CategoryRepository extends JpaRepository<Category, Long> {
}
package fun.asem.biograph.webapp.repository;

import fun.asem.biograph.webapp.domain.Event;
```

```
import org.springframework.data.jpa.repository.JpaRepository;

public interface EventRepository extends JpaRepository<Event, Long> {
}
package fun.asem.biograph.webapp.repository;

import fun.asem.biograph.webapp.domain.Grant;
import fun.asem.biograph.webapp.domain.User;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface GrantRepository extends JpaRepository<Grant, Long> {
   List<Grant>      findAllByUserAndAccessTypeAndAttributeIsNotNull(User      user,      Grant.AccessType
accessType);

   List<Grant>      findAllByUserAndAccessTypeAndCategoryIsNotNull(User      user,      Grant.AccessType
accessType);

   List<Grant> findAllByUserAndAccessTypeAndTagIsNotNull(User user, Grant.AccessType accessType);

   List<Grant> findAllByUserAndAccessTypeAndEventIsNotNull(User user, Grant.AccessType accessType);
}
package fun.asem.biograph.webapp.repository;

import fun.asem.biograph.webapp.domain.Tag;
import org.springframework.data.jpa.repository.JpaRepository;

public interface TagRepository extends JpaRepository<Tag, Long> {
}
package fun.asem.biograph.webapp.repository;

import fun.asem.biograph.webapp.domain.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.Optional;

@Repository
```

```
public interface UserRepository extends JpaRepository<User, Long> {
    boolean existsByCurrentEmail(String currentEmail);

    boolean existsByNickname(String nickname);

    Optional<User> findByCurrentEmail(String currentEmail);

    User getByCurrentEmail(String currentEmail);
}
.root {
}

.blurred {
  filter: blur(3px);
}

.progress-bar {
  position: fixed;
  bottom: 0;
  left: 0;
  width: 100vw;
  filter: blur(0px);
}

.progress-spinner {
  position: fixed;
  top: 49vh;
  left: 49vw;
  filter: blur(0px);
}
@color-gray: #ADADAD;

@angular-border-gray: #E0E0E0;
.login-form {
  text-align: center;
  margin-top: 10px;
}

.login-form-field {
```

```
  width: 600px;
  display: block;
  margin: auto;
  margin-top: 15px;
}

.divider-line {
  width: 60vw;
  margin: 10px auto;
}

.validation-error {
  width: 300px;
  margin-left: auto;
  margin-right: auto;
  margin-bottom: 21px;
}

@media screen and (max-device-width: 700px) {
  .login-form-field {
    width: 80vw;
  }

  .divider-line {
    width: 100vw;
  }

  .validation-error {
    width: 70vw;
  }
}

.text {
  font-style: normal;
  font-variant-ligatures: normal;
  font-variant-caps: normal;
  font-variant-numeric: normal;
  font-variant-east-asian: normal;
  font-weight: 500;
```

```
  font-stretch: normal;
  font-family: Roboto, "Helvetica Neue", sans-serif;
}

.normal-y-margin {
  margin-top: @normal-margin;
  margin-bottom: @normal-margin;
}

.small-y-margin {
  margin-top: @small-margin;
  margin-bottom: @small-margin;
}

.extra-small-y-margin {
  margin-top: @extra-small-margin;
  margin-bottom: @extra-small-margin;
}



.text-center {
  text-align: center;
}

.text-normal {
  .text;
  font-size: 50px;
}

.header {
  .text;
  text-align: center;
}

.block-centered-button {
  margin: auto;
  display: block;
}
```

```
.dialog-actions {
  display: flex;
  flex-flow: row wrap;
  justify-content: space-around;
}

.small-form-field {
  width: ~'min(210px, 70vw)';
  margin: auto;
}

.normal-form-field {
  width: 85%;
  margin: auto;
}

.rounded-color-picker {
  @picker-diameter: 25px;
  @picker-radius: @picker-diameter / 2;
  width: @picker-diameter;
  height: @picker-diameter;
  border-radius: @picker-radius;
  border: solid transparent;
}

@large-margin: 60px;
@big-margin: 40px;
@normal-margin: 20px;
@small-margin: 10px;
@extra-small-margin: 5px;
@import "../../common";

.container {
  text-align: center;
}

.attributes-header {
  .header;
  margin-top: @normal-margin;
```

```
}

.attributes-table {
  width: 100%;
  margin-top: @normal-margin;
}

::ng-deep .mat-sort-header-container {
  justify-content: center;
  /* The missing piece */
  margin-right: -18px;
}
@import "../../../common";

.new-attribute-form {
  text-align: center;
  padding-left: 15px;
  padding-right: 15px;
}

.attribute-dialog-form-field {
  width: 100%;
  margin: 10px auto auto;

  &__left-aligned {
    margin: 10px 0 auto;
    display: inline-block;
  }
}

.attribute-description-input {
  height: 40px;
}

.constraints-header {
  margin-top: @small-margin;
  font-weight: 500;
  letter-spacing: 0.03em;
}
```

```
.attribute-type-definition {
  display: flex;
  flex-flow: row wrap;
  justify-content: space-evenly;
  align-items: flex-end;
}

.short-number-form-field {
  width: 60px;
  margin-left: 10px;
  margin-right: 10px;
}

.centered-label {
  margin: auto;
}
@import "../../common";

.container {
  text-align: center;
}

.categories-header {
  .header;
  margin-top: @normal-margin;
}

.categories-table {
  width: 100%;
  margin-top: @normal-margin;
}

.mat-cell {
  text-align: center;
}

.create-category-button {
  display: block;
```

```
  margin: auto;
}

::ng-deep .mat-sort-header-container {
  justify-content: center;
  /* The missing piece */
  margin-right: -18px;
}

.search-bar {
  margin: auto;
}

.category-color-picker {
  @picker-diameter: 25px;
  @picker-radius: @picker-diameter / 2;
  width: @picker-diameter;
  height: @picker-diameter;
  border-radius: @picker-radius;
  border: solid transparent;
}
@import "../../../common";

.newCategoryDialogContainer {
  text-align: center;
}

.new-category-form {
  text-align: center;
}

.dialog-actions {
  display: flex;
  flex-flow: row wrap;
  justify-content: space-around;
}

.category-dialog-form-field {
  width: 85%;
```

```
  margin: auto;
}

.category-description-input {
  height: 40px;
}

.color-form-field {
  .category-dialog-form-field;
  margin-top: @normal-margin;
}

::ng-deep .arrow {
  visibility: hidden;
}

.attributes-header {
  margin-top: @small-margin;
  font-weight: 500;
  letter-spacing: 0.03em;
}

.attributes-table {
  width: 100%;
}

th {
  .text-center;
}

.add-attribute-button {
  display: block;
  margin: auto auto @small-margin;
}

.category-color-picker {
  .rounded-color-picker;
}
@pink-color: #ff4081;
```

```
@gray-color: #9b9b9b;


::ng-deep {
  .custom-slider .ng5-slider .ng5-slider-bar {
    background: @gray-color;
    height: 2px;
  }

  .custom-slider .ng5-slider .ng5-slider-selection {
    background: @pink-color;
  }

  .custom-slider .ng5-slider .ng5-slider-pointer {
    width: 8px;
    height: 16px;
    top: auto; /* to remove the default positioning */
    bottom: 0;
    background-color: @pink-color;
    border-top-left-radius: 3px;
    border-top-right-radius: 3px;
  }

  .custom-slider .ng5-slider .ng5-slider-pointer:after {
    display: none;
  }

  .custom-slider .ng5-slider .ng5-slider-bubble {
    bottom: 14px;
    font-size: 13px;
  }

  .custom-slider .ng5-slider .ng5-slider-limit {
    font-weight: bold;
    color: orange;
  }

}
@import "../../common";
```

```
@import "../../colors";

.event-card {
  width: ~"min(300px, 90vw)";
  max-width: 300px;
  height: 300px;
  border-top: thick solid;
  display: flex;
  flex-flow: column nowrap;
  justify-content: flex-start;
  align-items: center;
  align-content: center;

  &__header {
    display: flex;
    flex-flow: column wrap;
    justify-content: center;
    align-content: center;
    align-items: center;
  }

  &__description-content {
    width: 100%;
    border-top: thin solid @angular-border-gray;
    border-bottom: thin solid @angular-border-gray;
    flex-grow: 1;
    margin-bottom: @small-margin;
  }

  &__chip-list {
    margin-bottom: @small-margin;
  }

  &__chip-list:focus {
    outline: none;
  }

  &__content {
    //padding-top: @small-margin;
```

```
    margin-bottom: 1px;
  }

  &__tag-chip {
    border-right: 12px solid;
    margin: @extra-small-margin;
    padding: 1px 7px;
    border-radius: 16px;
    background-color: #E0E0E0;
    font-size: 10px;
    font-weight: 500;
  }

  &__tag-chip:focus {
    outline: none;
  }
}

.event-card-actions {
  width: 100%;
  padding-top: @extra-small-margin;
  display: flex;
  flex-flow: row wrap;
  justify-content: space-evenly;
  align-items: center;
  align-content: center;

  &__stub {
    flex-grow: 1;
  }
}

.event-card-action-button {
  margin-left: @normal-margin;
  margin-right: @normal-margin;
}

.event-card-action-button:focus {
  outline: none;
```

```
}
@import "../../../common";
@import "../../../colors";


.event-form {
  text-align: center;

  &__form-field {
    .normal-form-field;
  }

  &__small-form-field {
    .small-form-field;
  }

  &__extra-small-form-field {
    width: ~'min(90px, 25vw)';
    margin: auto;
  }

  &__time-bound-form-field {
    margin-top: 5px;
    margin-left: 10px;
  }

  &__divider-line {
    margin: 10px auto;
    width: 85%;
  }

  &__description-input {
    height: 80px;
  }

  &__section-header {
    margin-top: @small-margin;
    font-weight: 500;
    letter-spacing: 0.03em;
```

```
  }

  &__event-detail-section {
    width: 85%;
    text-align: center;
    margin: auto;
    margin-bottom: @extra-small-margin;

    &__text {
      color: black;
    }

  }

  &__attachment-section {
    margin-top: @small-margin;
  }

  &__parameters-section {
    display: flex;
    flex-flow: column nowrap;
    justify-content: center;
    align-items: center;
    align-content: center;

    &__parameter {
      margin-top: @small-margin;
    }
  }
}

.flexbox-row {
  display: flex;
  flex-flow: row wrap;
  justify-content: center;
  align-items: center;
  align-content: space-between;
}
```

```
:host ::ng-deep .attachment-form-drop-zone {
  border: medium dashed @color-gray;
  height: 100px;
}

:host ::ng-deep .attachment-form-area-content {
  height: 100%;
  display: flex;
  flex-flow: column nowrap;
  justify-content: space-around;
  align-items: center;
}

:host ::ng-deep .attachment-form-browse-button {
  border: 1px solid rgba(0, 0, 0, .12);
  padding: 0 15px;
  line-height: 34px;
  background: transparent;
  border-radius: 5px;
  font-family: Roboto, Helvetica Neue, sans-serif;
  font-size: 14px;
  font-weight: 500;
  display: block;
  margin-bottom: 10px;
}

:host ::ng-deep .ngx-file-drop__drop-zone-label {
  font-family: Roboto, Helvetica Neue, sans-serif;
  font-size: 14px;
  font-weight: 500;
  line-height: 34px;
}
@import "../../../../colors";
@import "../../../../common";

.parameter-container {
  border: thin solid @color-gray;
  border-radius: 5px;
  background: transparent;
```

```
  width: ~'min(230px, 80vw)';
  padding: 5px 5px 10px 5px;
}

.parameter-form-field {
  .small-form-field;
}
@import "../../../../common";

.container {
  text-align: center;
}
@import "../../common";

.container {
  text-align: center;

  &__event-card {
    margin-bottom: @normal-margin;
    margin-right: @normal-margin;
  }

  &__divider {
    margin-top: @normal-margin;
    margin-bottom: @normal-margin;
  }
}

.attributes-header {
  .header;
  margin-top: @normal-margin;
}

.events-section {
  display: flex;
  flex-flow: row wrap;
  justify-content: center;
  align-content: center;
  align-items: center;
```

```
}
.footer {
  height: 7vh;
  background-color: grey;

  position: fixed;
  bottom: 0;
  width: 100%;
}
@import "../../common";
@import "../../colors";

.home-container {
  text-align: center;
  padding-top: @normal-margin;

  &__greeting-header {
    .text-center;
  }

  &__greeting {
    display: flex;
    flex-flow: column nowrap;
    justify-content: flex-start;
    align-items: center;
    align-content: center;

    margin-top: @normal-margin;
    margin-left: auto;
    margin-right: auto;
    width: ~"min(600px, 90vw)";
    padding: 10px;

    &__logo {
      width: 80%;
    }

    &__list {
      text-align: left;
```

```
    }
  }
}
@import "../../common";
.sidenav-container {
  height: 100%;

  &__content {
    height: 100vh;
  }
}

.sidenav {
  width: 200px;
}

.sidenav .mat-toolbar {
  background: inherit;
}

.mat-toolbar.mat-primary {
  position: sticky;
  top: 0;
  z-index: 1;
}

.center {
  text-align: center;
}

.app-router-outlet {
  margin: auto;
  width: 90vw;
}

@import "../../common";

.not-found-container {
  text-align: center;
```

```
  &__header {
    margin-top: @normal-margin;
    margin-bottom: @normal-margin;
  }

  &__image {
    width: ~"min(600px, 80vw)";
  }
}
@import "../../common";
.search-form {
  min-width: 150px;
  max-width: 500px;
  width: 100%;
}

.search-form-field {
  width: 500px;
}

@media screen and (max-device-width: 700px) {
  .search-form-field {
    width: 80vw;
  }
}

@import "../../../common";


.pie-chart-root {
  display: flex;
  flex-flow: row wrap;
  justify-content: center;
  align-content: center;
  align-items: center;

  &__filters-bar {
    //border: 1px solid gray;
```

```
    width: ~"min(400px, 90vw)";
    height: 200px;

    &__header {
      font-family: Roboto, "Helvetica Neue", sans-serif;
      font-size: 14px;
      font-weight: 500;
      color: rgba(0, 0, 0, .87);
    }

    &__slider {
      margin-top: @large-margin;
    }
  }

  &__canvas {
    width: 700px;
  }
}
@import "../../../common";

.metrics-monitoring-root {
  text-align: center;
  width: 90vw;
  height: 80vh;

  &__line-chart {
    width: 100%;
    height: 100%;
  }

  &__content {
    margin: auto @normal-margin;

    &__config-bar {
      margin: auto;
      width: 40vw;
      height: 250px;
```

```
    &__header {
      .text;
    }

    &__main-section {
      display: grid;
      grid-template-columns: 50% 50%;
      justify-items: center;
      align-items: start;

      &__metric-selector {
        grid-column: 1;
      }

      &__metric-configuration {
        grid-column: 2;
        display: flex;
        flex-flow: column nowrap;
        justify-content: center;
        align-items: flex-start;
        align-content: flex-start;
      }
    }
  }


}

/* &__configuration-bar {
    width: 14vw;
    height: 400px;
    float: right;
    position: relative;
    bottom: 400px;
    &__header {
      .text;
    }
  }*/
}
```

```
.statistics-page-container {
  text-align: center;
}

.statistic-tab-bar {
  margin: auto;
  width: 100%;
  height: 90vh;

}

.generic-tab {
  width: 90vw;
  height: 90vh;
}

.metrics-monitoring-tab {
  .generic-tab
}

:host ::ng-deep .mat-tab-labels {
  justify-content: center; /* align items in Main Axis */
}
.tab-bar {
  height: 7vh;
  background-color: grey;

  margin-top: 1vh;
}
@import "../../../common";

.tag-form {
  text-align: center;
  padding-left: 15px;
  padding-right: 15px;
}

.tag-dialog-form-field {
  .normal-form-field;
```

```
}

.tag-description-input {
  height: 40px;
}

.color-form-field {
  .tag-dialog-form-field;
  margin-top: @normal-margin;
}

// to hide arrow on color picker component
:host ::ng-deep .arrow {
  visibility: hidden;
}
@import "../../common";

.container {
  text-align: center;
}

.tags-header {
  .header;
  margin-top: @normal-margin;
}

.tags-table {
  width: 100%;
  margin-top: @normal-margin;
}

::ng-deep .mat-sort-header-container {
  justify-content: center;
  /* The missing piece */
  margin-right: -18px;
}
/* You can add global styles to this file, and also import other style files */

html, body {
```

```
  height: 100%;
}

body {
  margin: 0;
  font-family: Roboto, "Helvetica Neue", sans-serif;
}

@import "~ng-pick-datetime/assets/style/picker.min.css";
{
  "name": "biograph-frontend",
  "version": "0.0.0",
  "lockfileVersion": 1,
  "requires": true,
  "dependencies": {
    "@angular-devkit/architect": {
      "version": "0.901.5",
      "resolved": "https://registry.npmjs.org/@angular-devkit/architect/-/architect-0.901.5.tgz",
      "integrity": "sha512-
VO+8qBkaq54xAjdtvhEhQ86gZxS0V1wC9hGblw3O+XXri/euHky4811B2BbEylPy8/kRy5sUYcuwcyZrVxJ
2TQ==",
      "dev": true,
      "requires": {
        "@angular-devkit/core": "9.1.5",
        "rxjs": "6.5.4"
      },
      "dependencies": {
        "rxjs": {
          "version": "6.5.4",
          "resolved": "https://registry.npmjs.org/rxjs/-/rxjs-6.5.4.tgz",
          "integrity": "sha512-
naMQXcgEo3csAEGvw/NydRA0fuS2nDZJiw1YUWFKU7aPPAPGZEsD4Iimit96qwCieH6y614MCLYwdkr
Wx7z/7Q==",
          "dev": true,
          "requires": {
            "tslib": "^1.9.0"
          }
        }
      }
```

```
    },
    "@angular-devkit/build-angular": {
      "version": "0.901.5",
      "resolved": "https://registry.npmjs.org/@angular-devkit/build-angular/-/build-angular-0.901.5.tgz",
      "integrity": "sha512-XottEBXE7cmkx6LPu33lXJCSAlxFkIu2ErWvV1oy+La6aZEuoJVntxzIKLprJmTiiD/4IDDQLWwp4m+EC96kyg==",
      "dev": true,
      "requires": {
        "@angular-devkit/architect": "0.901.5",
        "@angular-devkit/build-optimizer": "0.901.5",
        "@angular-devkit/build-webpack": "0.901.5",
        "@angular-devkit/core": "9.1.5",
        "@babel/core": "7.9.0",
        "@babel/generator": "7.9.3",
        "@babel/preset-env": "7.9.0",
        "@babel/template": "7.8.6",
        "@jsdevtools/coverage-istanbul-loader": "3.0.3",
        "@ngtools/webpack": "9.1.5",
        "ajv": "6.12.0",
        "autoprefixer": "9.7.4",
        "babel-loader": "8.0.6",
        "browserslist": "^4.9.1",
        "cacache": "15.0.0",
        "caniuse-lite": "^1.0.30001032",
        "circular-dependency-plugin": "5.2.0",
        "copy-webpack-plugin": "5.1.1",
        "core-js": "3.6.4",
        "css-loader": "3.5.1",
        "cssnano": "4.1.10",
        "file-loader": "6.0.0",
        "find-cache-dir": "3.3.1",
        "glob": "7.1.6",
        "jest-worker": "25.1.0",
        "karma-source-map-support": "1.4.0",
        "less": "3.11.1",
        "less-loader": "5.0.0",
        "license-webpack-plugin": "2.1.4",
        "loader-utils": "2.0.0",
```

```
    "mini-css-extract-plugin": "0.9.0",
    "minimatch": "3.0.4",
    "open": "7.0.3",
    "parse5": "4.0.0",
    "postcss": "7.0.27",
    "postcss-import": "12.0.1",
    "postcss-loader": "3.0.0",
    "raw-loader": "4.0.0",
    "regenerator-runtime": "0.13.5",
    "rimraf": "3.0.2",
    "rollup": "2.1.0",
    "rxjs": "6.5.4",
    "sass": "1.26.3",
    "sass-loader": "8.0.2",
    "semver": "7.1.3",
    "source-map": "0.7.3",
    "source-map-loader": "0.2.4",
    "speed-measure-webpack-plugin": "1.3.1",
    "style-loader": "1.1.3",
    "stylus": "0.54.7",
    "stylus-loader": "3.0.2",
    "terser": "4.6.10",
    "terser-webpack-plugin": "2.3.5",
    "tree-kill": "1.2.2",
    "webpack": "4.42.0",
    "webpack-dev-middleware": "3.7.2",
    "webpack-dev-server": "3.10.3",
    "webpack-merge": "4.2.2",
    "webpack-sources": "1.4.3",
    "webpack-subresource-integrity": "1.4.0",
    "worker-plugin": "4.0.3"
  },
  "dependencies": {
    "rxjs": {
      "version": "6.5.4",
      "resolved": "https://registry.npmjs.org/rxjs/-/rxjs-6.5.4.tgz",
      "integrity": "sha512-
naMQXcgEo3csAEGvw/NydRA0fuS2nDZJiw1YUWFKU7aPPAPGZEsD4Iimit96qwCieH6y614MCLYwdkr
Wx7z/7Q==",
```

```
        "dev": true,
        "requires": {
          "tslib": "^1.9.0"
        }
      }
    }
  },
  "@angular-devkit/build-optimizer": {
    "version": "0.901.5",
    "resolved": "https://registry.npmjs.org/@angular-devkit/build-optimizer/-/build-optimizer-0.901.5.tgz",
    "integrity": "sha512-
xmAMvLMSa8BvqlZ0wsC37Qop/7/pEaQRKLeowC3CCI3jiYDF10Tihar+Hjc04NVSal8ZBP9/+Gp3Yr0x61
HcFA==",
    "dev": true,
    "requires": {
      "loader-utils": "2.0.0",
      "source-map": "0.7.3",
      "tslib": "1.11.1",
      "typescript": "3.6.5",
      "webpack-sources": "1.4.3"
    },
    "dependencies": {
      "tslib": {
        "version": "1.11.1",
        "resolved": "https://registry.npmjs.org/tslib/-/tslib-1.11.1.tgz",
        "integrity": "sha512-
aZW88SY8kQbU7gpV19lN24LtXh/yD4ZZg6qieAJDDg+YBsJcSmLGK9QpnUjAKVG/xefmvJGd1WUmfpT
/g6AJGA==",
        "dev": true
      },
      "typescript": {
        "version": "3.6.5",
        "resolved": "https://registry.npmjs.org/typescript/-/typescript-3.6.5.tgz",
        "integrity": "sha512-
BEjlc0Z06ORZKbtcxGrIvvwYs5hAnuo6TKdNFL55frVDlB+na3z5bsLhFaIxmT+dPWgBIjMo6aNnTOgHH
mHgiQ==",
        "dev": true
      }
    }
```

```
    },
    "@angular-devkit/build-webpack": {
      "version": "0.901.5",
      "resolved": "https://registry.npmjs.org/@angular-devkit/build-webpack/-/build-webpack-0.901.5.tgz",
      "integrity": "sha512-
4HHOFFRu3yUBe0otaDHh4PID99XvnBJj7hYzvbB5BvP0ULif4/W0aoU2STnern4lH4pJmrtqYawBZ84ETX
N/Ag==",
      "dev": true,
      "requires": {
        "@angular-devkit/architect": "0.901.5",
        "@angular-devkit/core": "9.1.5",
        "rxjs": "6.5.4"
      },
      "dependencies": {
        "rxjs": {
          "version": "6.5.4",
          "resolved": "https://registry.npmjs.org/rxjs/-/rxjs-6.5.4.tgz",
          "integrity": "sha512-
naMQXcgEo3csAEGvw/NydRA0fuS2nDZJiw1YUWFKU7aPPAPGZEsD4Iimit96qwCieH6y614MCLYwdkr
Wx7z/7Q==",
          "dev": true,
          "requires": {
            "tslib": "^1.9.0"
          }
        }
      }
    },
    "@angular-devkit/core": {
      "version": "9.1.5",
      "resolved": "https://registry.npmjs.org/@angular-devkit/core/-/core-9.1.5.tgz",
      "integrity": "sha512-
i0BJ6Ad3bcDE6e4Ev9Flbw7P0bz9p94FDVfEOPGBTrbJQZSqOm3CqaH2y5LGfl7acSPUi54hK481h1QRUF
BkTQ==",
      "dev": true,
      "requires": {
        "ajv": "6.12.0",
        "fast-json-stable-stringify": "2.1.0",
        "magic-string": "0.25.7",
        "rxjs": "6.5.4",
```

```
      "source-map": "0.7.3"
    },
    "dependencies": {
      "rxjs": {
        "version": "6.5.4",
        "resolved": "https://registry.npmjs.org/rxjs/-/rxjs-6.5.4.tgz",
        "integrity": "sha512-
naMQXcgEo3csAEGvw/NydRA0fuS2nDZJiw1YUWFKU7aPPAPGZEsD4Iimit96qwCieH6y614MCLYwdkr
Wx7z/7Q==",
        "dev": true,
        "requires": {
          "tslib": "^1.9.0"
        }
      }
    }
  },
  "@angular-devkit/schematics": {
    "version": "9.1.5",
    "resolved": "https://registry.npmjs.org/@angular-devkit/schematics/-/schematics-9.1.5.tgz",
    "integrity": "sha512-
nkNiGrVl7xLaYYj/oT1wOBowa4Driv2f4abn78AJI/pd/EXA45G/rI9gO/kEG8lHn+FAMQedaywX9N4JDOx
CGg==",
    "dev": true,
    "requires": {
      "@angular-devkit/core": "9.1.5",
      "ora": "4.0.3",
      "rxjs": "6.5.4"
    },
    "dependencies": {
      "rxjs": {
        "version": "6.5.4",
        "resolved": "https://registry.npmjs.org/rxjs/-/rxjs-6.5.4.tgz",
        "integrity": "sha512-
naMQXcgEo3csAEGvw/NydRA0fuS2nDZJiw1YUWFKU7aPPAPGZEsD4Iimit96qwCieH6y614MCLYwdkr
Wx7z/7Q==",
        "dev": true,
        "requires": {
          "tslib": "^1.9.0"
        }
```

```
      }
     }
    },
    "@angular/animations": {
     "version": "9.1.6",
     "resolved": "https://registry.npmjs.org/@angular/animations/-/animations-9.1.6.tgz",
     "integrity": "sha512-
7Pp7aqNNcH4fu1BnOVpvqJJHjE7RZ5K1oD396OWCh35pgpLowLSpFjhbVhzGrcAuxHyKnnHSX3etLn2h
DaHxmQ=="
    },
    "@angular/cdk": {
     "version": "9.2.3",
     "resolved": "https://registry.npmjs.org/@angular/cdk/-/cdk-9.2.3.tgz",
     "integrity": "sha512-
tQr/yt8GNGsZ/DTT+PMq7XdRmL56hwVCyf8F12JQAawutSLfTfMb+S1lpN7L/0Pb/L5JBuCfG2HmXK7vH
o02gw==",
     "requires": {
      "parse5": "^5.0.0"
     },
     "dependencies": {
      "parse5": {
       "version": "5.1.1",
       "resolved": "https://registry.npmjs.org/parse5/-/parse5-5.1.1.tgz",
       "integrity": "sha512-
ugq4DFI0Ptb+WWjAdOK16+u/nHfiIrcE+sh8kZMaM0WllQKLI9rOUq6c2b7cwPkXdzfQESqvoqK6ug7U/Y
yzug==",
       "optional": true
      }
     }
    },
    "@angular/cli": {
     "version": "9.1.5",
     "resolved": "https://registry.npmjs.org/@angular/cli/-/cli-9.1.5.tgz",
     "integrity": "sha512-
Decn0p+s0SLc5p2GN7k8dUrOY43hBuCz8PxueW0gEln7NjlpdmHk+6JQn0ZRl2/2Dbk+eTGzPXyCxGwqs
W3jjw==",
     "dev": true,
     "requires": {
      "@angular-devkit/architect": "0.901.5",
```

```
    "@angular-devkit/core": "9.1.5",
    "@angular-devkit/schematics": "9.1.5",
    "@schematics/angular": "9.1.5",
    "@schematics/update": "0.901.5",
    "@yarnpkg/lockfile": "1.1.0",
    "ansi-colors": "4.1.1",
    "debug": "4.1.1",
    "ini": "1.3.5",
    "inquirer": "7.1.0",
    "npm-package-arg": "8.0.1",
    "npm-pick-manifest": "6.0.0",
    "open": "7.0.3",
    "pacote": "9.5.12",
    "read-package-tree": "5.3.1",
    "rimraf": "3.0.2",
    "semver": "7.1.3",
    "symbol-observable": "1.2.0",
    "universal-analytics": "0.4.20",
    "uuid": "7.0.2"
  },
  "dependencies": {
    "ansi-colors": {
      "version": "4.1.1",
      "resolved": "https://registry.npmjs.org/ansi-colors/-/ansi-colors-4.1.1.tgz",
      "integrity": "sha512-
JoX0apGbHaUJBNl6yF+p6JAFYZ666/hhCGKN5t9QFjbJQKUU/g8MNbFDbvfrgKXvI1QpZplPOnwIo99lX/
AAmA==",
      "dev": true
    },
    "uuid": {
      "version": "7.0.2",
      "resolved": "https://registry.npmjs.org/uuid/-/uuid-7.0.2.tgz",
      "integrity": "sha512-
vy9V/+pKG+5ZTYKf+VcphF5Oc6EFiu3W8Nv3P3zIh0EqVI80ZxOzuPfe9EHjkFNvf8+xuTHVeei4Drydlx4z
jw==",
      "dev": true
    }
  }
},
```

```
    "@angular/common": {
      "version": "9.1.6",
      "resolved": "https://registry.npmjs.org/@angular/common/-/common-9.1.6.tgz",
      "integrity": "sha512-
L9vw//wE+8QcSArOA411uJ68znnszCiPrbzSBV0BRZeadc7X68MwANA9qjtiTWZx5Xh9pNfHHwsCUyv2l
UeinQ=="
    },
    "@angular/compiler": {
      "version": "9.1.6",
      "resolved": "https://registry.npmjs.org/@angular/compiler/-/compiler-9.1.6.tgz",
      "integrity": "sha512-
tHOQEjWuWqSkrk6/vI6BK7uJnpAyFajCXPW37rH5xspF/aMbetrpoOiGBNUQg/HLaFh04nOAnnFntjLkW0
ooaA=="
    },
    "@angular/compiler-cli": {
      "version": "9.1.6",
      "resolved": "https://registry.npmjs.org/@angular/compiler-cli/-/compiler-cli-9.1.6.tgz",
      "integrity": "sha512-
1b7m9tvSQJE4y95wNCdi34MORcsmXPC1vaylYlzChVM2et9Y2eKHYcRs+4g329j66irLBGpS/7cgTT2pgJ+
IbA==",
      "dev": true,
      "requires": {
        "canonical-path": "1.0.0",
        "chokidar": "^3.0.0",
        "convert-source-map": "^1.5.1",
        "dependency-graph": "^0.7.2",
        "fs-extra": "4.0.2",
        "magic-string": "^0.25.0",
        "minimist": "^1.2.0",
        "reflect-metadata": "^0.1.2",
        "semver": "^6.3.0",
        "source-map": "^0.6.1",
        "sourcemap-codec": "^1.4.8",
        "yargs": "15.3.0"
      },
      "dependencies": {
        "ansi-regex": {
          "version": "5.0.0",
          "resolved": "https://registry.npmjs.org/ansi-regex/-/ansi-regex-5.0.0.tgz",
```

```
    "integrity": "sha512-bY6fj56OUQ0hU1KjFNDQuJFezqKdrAyFdIevADiqrWHwSlbmBNMHp5ak2f40Pm8JTFyM2mqxkG6ngkHO11f/lg==",
    "dev": true
   },
   "ansi-styles": {
    "version": "4.2.1",
    "resolved": "https://registry.npmjs.org/ansi-styles/-/ansi-styles-4.2.1.tgz",
    "integrity": "sha512-9VGjrMsG1vePxcSweQsN20KY/c4zN0h9fLjqAbwbPfahM3t+NL+M9HC8xeXG2I8pX5NoamTGNuomEUFI7fcUjA==",
    "dev": true,
    "requires": {
     "@types/color-name": "^1.1.1",
     "color-convert": "^2.0.1"
    }
   },
   "cliui": {
    "version": "6.0.0",
    "resolved": "https://registry.npmjs.org/cliui/-/cliui-6.0.0.tgz",
    "integrity": "sha512-t6wbgtoCXvAzst7QgXxJYqPt0usEfbgQdftEPbLL/cvv6HPE5VgvqCuAIDR0NgU52ds6rFwqrgakNLrHEjCbrQ==",
    "dev": true,
    "requires": {
     "string-width": "^4.2.0",
     "strip-ansi": "^6.0.0",
     "wrap-ansi": "^6.2.0"
    }
   },
   "color-convert": {
    "version": "2.0.1",
    "resolved": "https://registry.npmjs.org/color-convert/-/color-convert-2.0.1.tgz",
    "integrity": "sha512-RRECPsj7iu/xb5oKYcsFHSppFNnsj/52OVTRKb4zP5onXwVF3zVmmToNcOfGC+CRDpfK/U584fMg38ZHCaElKQ==",
    "dev": true,
    "requires": {
     "color-name": "~1.1.4"
```

```
      }
    },
    "color-name": {
      "version": "1.1.4",
      "resolved": "https://registry.npmjs.org/color-name/-/color-name-1.1.4.tgz",
      "integrity": "sha512-dOy+3AuW3a2wNbZHIuMZpTcgjGuLU/uBL/ubcZF9OXbDo8ff4O8yVp5Bf0efS8uEoYo5q4Fx7dY9OgQGXgAsQA==",
      "dev": true
    },
    "find-up": {
      "version": "4.1.0",
      "resolved": "https://registry.npmjs.org/find-up/-/find-up-4.1.0.tgz",
      "integrity": "sha512-PpOwAdQ/YlXQ2vj8a3h8IipDuYRi3wceVQQGYWxNINccq40Anw7BlsEXCMbt1Zt+OLA6Fq9suIpIWD0OsnISlw==",
      "dev": true,
      "requires": {
        "locate-path": "^5.0.0",
        "path-exists": "^4.0.0"
      }
    },
    "get-caller-file": {
      "version": "2.0.5",
      "resolved": "https://registry.npmjs.org/get-caller-file/-/get-caller-file-2.0.5.tgz",
      "integrity": "sha512-DyFP3BM/3YHTQOCUL/w0OZHR0lpKeGrxotcHWcqNEdnltqFwXVfhEBQ94eIo34AfQpo0rGki4cyIiftY06h2Fg==",
      "dev": true
    },
    "is-fullwidth-code-point": {
      "version": "3.0.0",
      "resolved": "https://registry.npmjs.org/is-fullwidth-code-point/-/is-fullwidth-code-point-3.0.0.tgz",
      "integrity": "sha512-zymm5+u+sCsSWyD9qNaejV3DFvhCKclKdizYaJUuHA83RLjb7nSuGnddCHGv0hk+KY7BMAlsWeK4Ueg6EV6XQg==",
      "dev": true
    },
    "locate-path": {
```

```
      "version": "5.0.0",
      "resolved": "https://registry.npmjs.org/locate-path/-/locate-path-5.0.0.tgz",
      "integrity": "sha512-t7hw9pI+WvuwNJXwk5zVHpyhIqzg2qTlklJOf0mVxGSbe3Fp2VieZcduNYjaLDoy6p9uGpQEGWG87WpMKlNq8g==",
      "dev": true,
      "requires": {
        "p-locate": "^4.1.0"
      }
    },
    "p-limit": {
      "version": "2.3.0",
      "resolved": "https://registry.npmjs.org/p-limit/-/p-limit-2.3.0.tgz",
      "integrity": "sha512-//88mFWSJx8lxCzwdAABTJL2MyWB12+eIY7MDL2SqLmAkeKU9qxRvWuSyTjm3FUmpBEMuFfckAIqEaVGUDxb6w==",
      "dev": true,
      "requires": {
        "p-try": "^2.0.0"
      }
    },
    "p-locate": {
      "version": "4.1.0",
      "resolved": "https://registry.npmjs.org/p-locate/-/p-locate-4.1.0.tgz",
      "integrity": "sha512-R79ZZ/0wAxKGu3oYMlz8jy/kbhsNrS7SKZ7PxEHBgJ5+F2mtFW2fK2cOtBh1cHYkQsbzFV7I+EoRKe6Yt0oK7A==",
      "dev": true,
      "requires": {
        "p-limit": "^2.2.0"
      }
```

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Biograph</title>
```

```
  <base href="/">
  <meta content="width=device-width, initial-scale=1" name="viewport">
  <link href="assets/img/favicon.ico" rel="icon" type="image/x-icon">
  <link href="https://fonts.googleapis.com/css?family=Roboto:300,400,500&display=swap" rel="stylesheet">
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
</head>
<body class="mat-typography">
<app-root></app-root>
</body>
</html>
import {Component, OnInit} from '@angular/core';
import {AppState, getProgressBarStatus, getProgressSpinnerStatus} from './store/app.state';
import {Store} from '@ngrx/store';
import {Observable} from 'rxjs';


@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.less']
})
export class AppComponent implements OnInit {
  title = 'biograph-frontend';
  isProgressBarVisible: Observable<boolean>;
  isProgressSpinnerVisible: Observable<boolean>;


  constructor(
    private store$: Store<AppState>,
```

```
  ) {
  }

  ngOnInit(): void {
    this.isProgressSpinnerVisible = this.store$.select(getProgressSpinnerStatus);
    this.isProgressBarVisible = this.store$.select(getProgressBarStatus);
  }



}
import {Component, OnInit} from '@angular/core';

@Component({
  selector: 'app-not-found',
  templateUrl: './not-found.component.html',
  styleUrls: ['./not-found.component.less']
})
export class NotFoundComponent implements OnInit {

  constructor() {
  }

  ngOnInit(): void {
  }

}
```

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

ВЕБ-ЗАСТОСУВАННЯ МОНІТОРИНГУ ЖИТТЄВИХ ПОДІЙ

Програма та методика тестування

КПІ.ІП-6326.045420.05.51

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Ковтунець О.В.

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____А.О. Семченко

Київ – 2020 року

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробувань, що описані в даному документі, є веб-застосування “Biograph”, що призначене для аналізу та структурованого зберігання інформації про життєві події. В рамках даного випробування проводиться дослідження серверної та клієнтська частини версії “v2.1” застосунку “Biograph”.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є:

- визначити ступінь відповідності реальної поведінки застосунку “Biograph” очікуваній поведінці, що описана у функціональних та нефункціональних вимогах до цього програмного продукту;
- прийняти рішення про придатність або непридатність поточної версії програмного продукту “Biograph” до промислової експлуатації.

3 МЕТОДИ ТЕСТУВАННЯ

Тестування проводиться для кожного тестового прецеденту (test case). Опис тестового прецеденту включає в себе:

- унікальний ідентифікатор прецеденту в рамках документу;
- текстовий опис прецеденту;
- передумови;
- постумови;
- основний сценарій проведення тестування прецеденту;
- очікуваний результат;
- отриманий результат;
- дату створення прецеденту.

Методологічним підходом, за яким проводиться тестування є “Тестування за стратегією чорного ящика”. Тестування проводиться для кожного тестового прецеденту. Спосіб ведення обліку результатів тестування за кожним тестовим прецедентом не регламентується даним документом. Вибір даного способу покладено на суб’єкт, що проводить тестування.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування програмного продукту “Biograph” проводиться з використанням програмних та технічних засобів, перелік яких наведено нижче.

Програмні засоби:

- браузер Google Chrome версії 81.0;
- браузер Mozilla Firefox версії 76.0.1;
- веб-застосування qase.io;
- операційна система Windows 10.

У склад технічних засобів входить персональний комп’ютер, що включає у себе:

- процесор Intel Core i5 з тактовою частотою, ГГц – 2, не менше;
- оперативну пам’ять об’ємом, Гб, 1, не менше;
- жорсткий або твердотільний накопичувач об’ємом, Гб, 25, не менше.

Для проведення випробувань розробник програмного продукту надає інсталяційну версії розробленого продукту.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

ВЕБ-ЗАСТОСУВАННЯ МОНІТОРИНГУ ЖИТТЄВИХ ПОДІЙ

Керівництво користувача

КПІ.ІП-6326.045420.06.34

“ПОГОДЖЕНО”

Керівник проєкту:

_____ О.В. Ковтунець

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____А.О. Семченко

Київ – 2020 року

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

1.1 Функціональне призначення

Функціональним призначенням програмного продукту “Biograph” є надання можливості користувачу зберігати свій життєвий досвід онлайн, та аналізувати збережені дані.

1.2 Експлуатаційне призначення

Програма призначена для експлуатації людьми, що активно контролюють своє життя або бажають покращити якість свого життя шляхом відстеження певних його параметрів.

1.3 Склад функцій

Програма забезпечує можливість виконання наведених нижче функцій:

- функції реєстрації в системі;
- функції авторизації в системі;
- функції створення тегу;
- функції створення метрики;
- функції створення категорії події;
- функції створення події;
- функції пошуку тегу;
- функції пошуку метрики;
- функції пошуку категорії;
- функції пошуку події;
- функції перегляду кругової діаграми, що показує частку подій за кожною категорією;
- функції перегляду кругової діаграми, що показує частку подій за кожним тегом;
- функції перегляду лінійного графіку зміни значень метрик у часі.

2 УМОВИ ВИКОНАННЯ ПРОГРАМИ

2.1 Кліматичні умови експлуатації

Кліматичні умови експлуатації, за яких повинні забезпечуватися задані характеристики та функції програмного забезпечення, повинні задовольняти вимогам, що висуваються до технічних засобів, необхідних для експлуатації програмного забезпечення.

2.2 Мінімальний склад технічних засобів

До складу технічних засобів повинен входити компьютер, що включає в себе:

- процесор Intel Core i5 з тактовою частотою, ГГц – 2, не менше;
- оперативну пам’ять об’ємом, Гб, 1, не менше;
- жорсткий або твердотільний накопичувач об’ємом, Гб, 25, не менше.

2.3 Мінімальний склад програмних засобів

Програмне забезпечення “Biograph” потребує наступний перелік програмних засобів:

- операційна система “Ubuntu Server” версії “16.04”;
- система керування базами даних “PostgreSQL” версії “12.3”;
- браузер “Google Chrome” версії “81.0”.

3 ВИКОНАННЯ ПРОГРАМИ

3.1 Початок роботи

Для початку роботи з програмним забезпеченням, оператор повинен відкрити браузер та ввести у адресний рядок ір-адресу або доменне ім’я серверу, на якому розгорнуто програмне забезпечення. Запуск браузера проводиться способами, детальна інформація про які наведена у керівництві користувача операційної системи. Для отримання IP-адреси або доменного імені серверу, на якому розгорнуто програмне забезпечення, оператору слід звернутись до суб'єкта, що проводив розгортання програмного забезпечення.

3.2 Використання програми

Перш ніж використовувати програму, користувачу необхідно виділити перелік ключових показників ефективності(в термінах даного програмного продукту - метрик). Після цього, користувач створює модель подій свого життя. Модель подій визначається переліком категорій подій.

Після визначення структури можливих життєвих подій, користувач вносить інформацію, про життєві події у систему. Частота внесення подій не регламентується даним програмним забезпеченням і залишається на розсуд користувача.

3.2.1 Реєстрація у системі

Для реєстрації у системі, необхідно натиснути кнопку, з графічним позначення меню. Після цього, зліва з’явиться бокове меню. У боковому меню, що з’явилося необхідно натиснути кнопку “Sign up”. Після цього, з’явиться форма реєстрації, у кожне поле якої необхідно ввести дані, відповідно до інструкції над кожною формою введення. Після введення даних, необхідно натиснути кнопку “Sign up”, що знаходиться у нижній частині форми. У разі,

якщо кнопка не розблокувалася – необхідно виконувати вказівки, що демонструє програма.

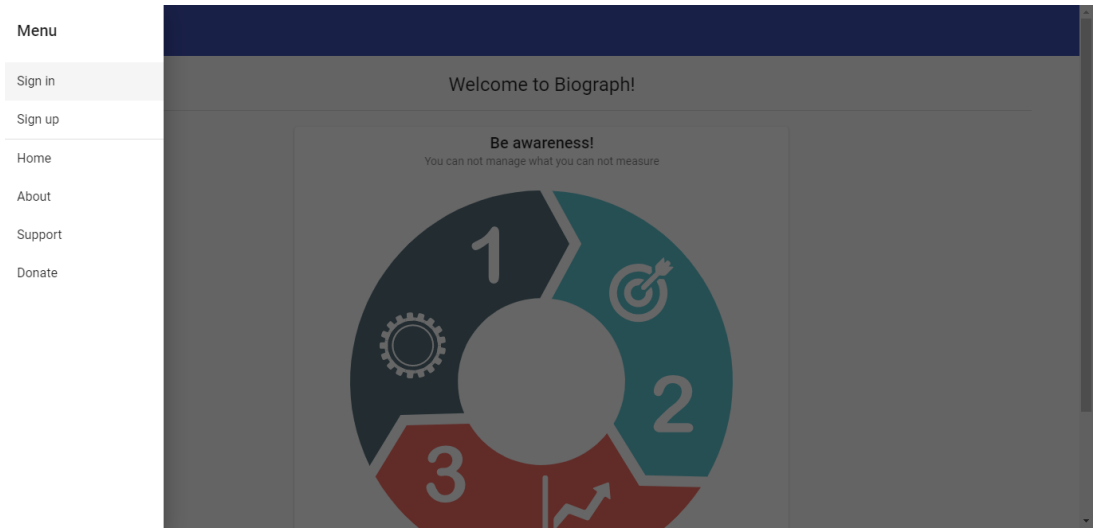


Рисунок 3.1 – Відкрите бокове меню застосунку

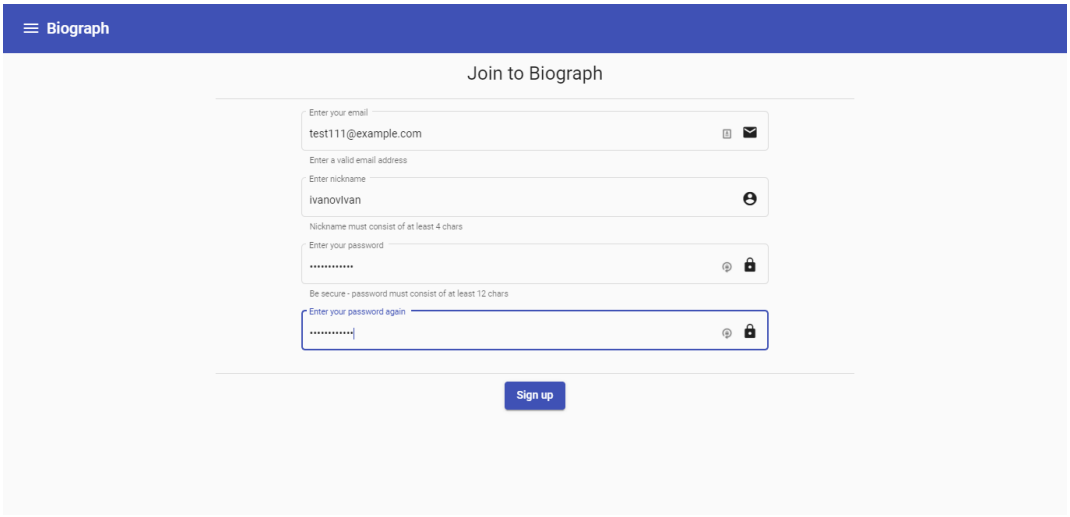


Рисунок 3.2 – Приклад заповненої форми реєстрації

3.2.2 Авторизація у системі

Для авторизації у системі, необхідно:

- відкрити бокове меню додатку;
- натиснути кнопку “Sign in”;
- у формі авторизації, що з'явилася, ввести логін та пароль;
- натиснути кнопку “Sign in”.

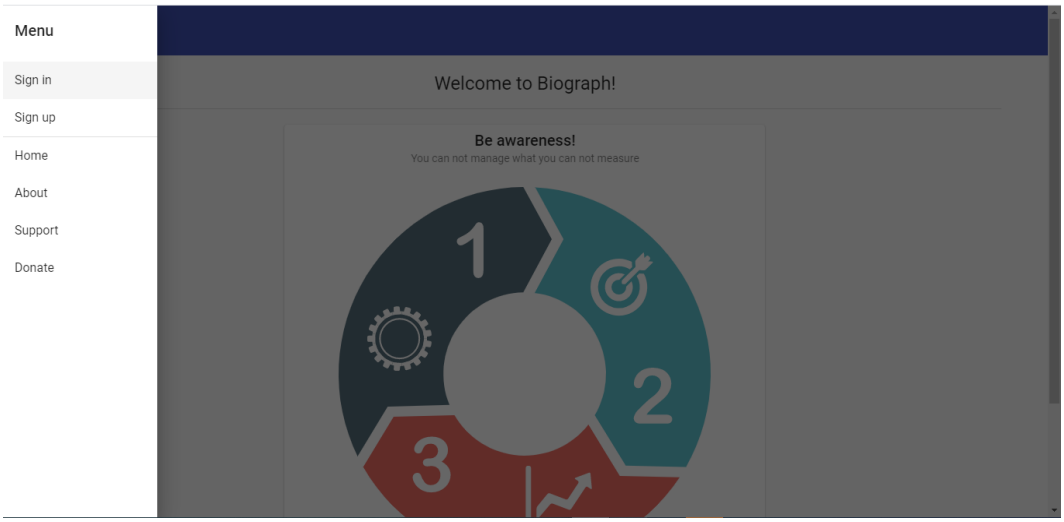


Рисунок 3.3 – Відкрите бокове меню, з виділеною кнопкою авторизації

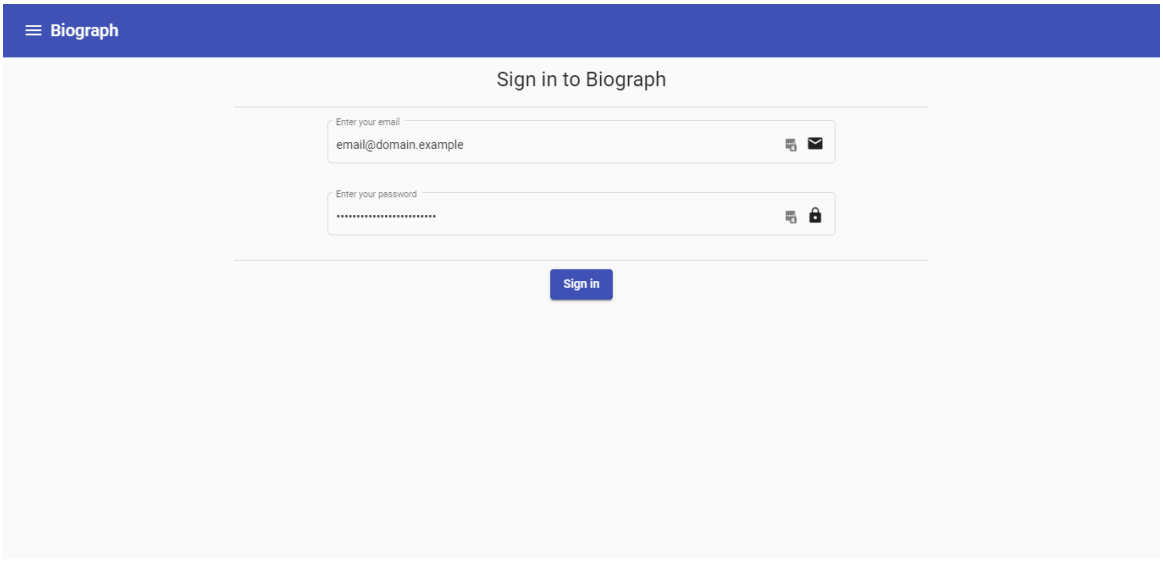


Рисунок 3.4 – Приклад заповненої форми авторизації

3.2.3 Створення тегу

Для створення нового тегу необхідно:

- відкрити бокове меню застосунку;
- натиснути кнопку “Tags”;
- на сторінці “Tags”, що відобразилася, натиснути кнопку “New”;
- у діалоговому вікні, що відкрилося, ввести необхідні поля форми;
- натиснути кнопку “Create”.

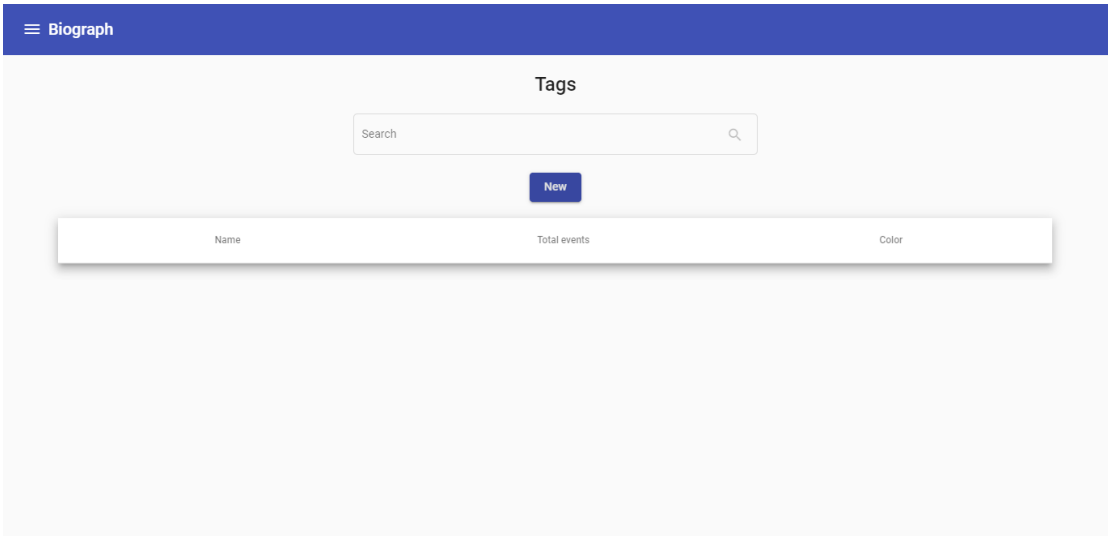


Рисунок 3.5 – Сторінка “Tags”

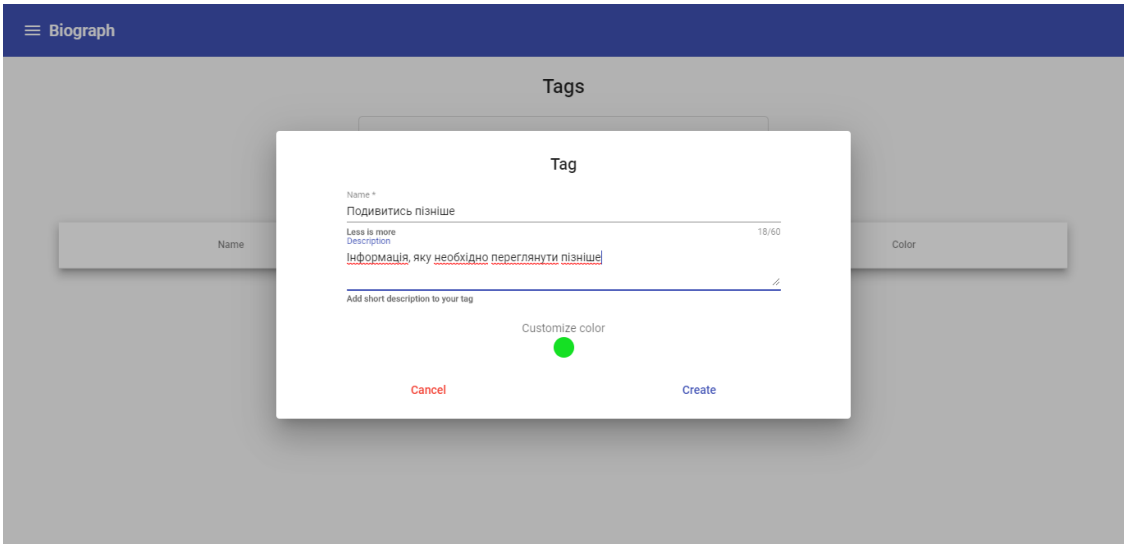


Рисунок 3.6 – Приклад заповненої форми створення тегу

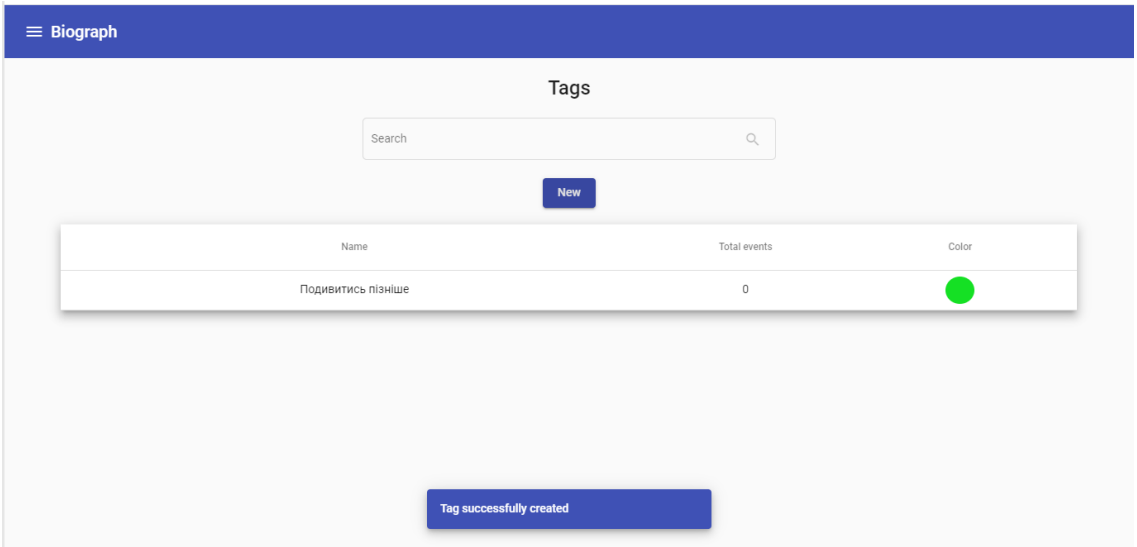


Рисунок 3.7 – Результат натискання кнопки “Create”

3.2.4 Створення метрики

Для створення метрики необхідно:

- відкрити бокове меню застосунку;
- натиснути кнопку “Metrics”;
- на сторінці “Metrics”, що відкрилась, натиснути кнопку “New”;
- у діалоговому вікні, що відкрилось, заповнити необхідні поля, відповідно до інструкцій, що демонтструє додаток;
- натиснути кнопку “Create”.

Додаток передбачає можливість створення метрик двох різних типів:

- числова метрика – метрика, що описує число. Можливі значення метрики можуть бути обмежені. Для цього потрібно обрати необхідне обмеження у полі введення “Constraint type”;
- номінальна метрика – метрика, що може набувати одного значення з переліку. Переліку можливих значень задається рядком, в якому кожне значення розділене комою.

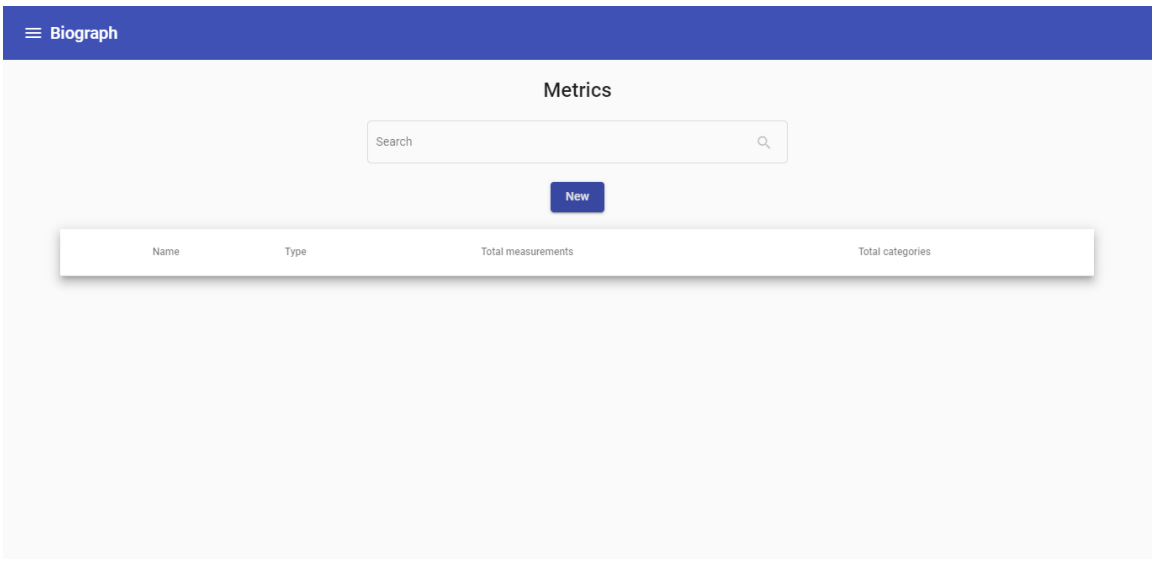


Рисунок 3.8 – Сторінка “Metrics”

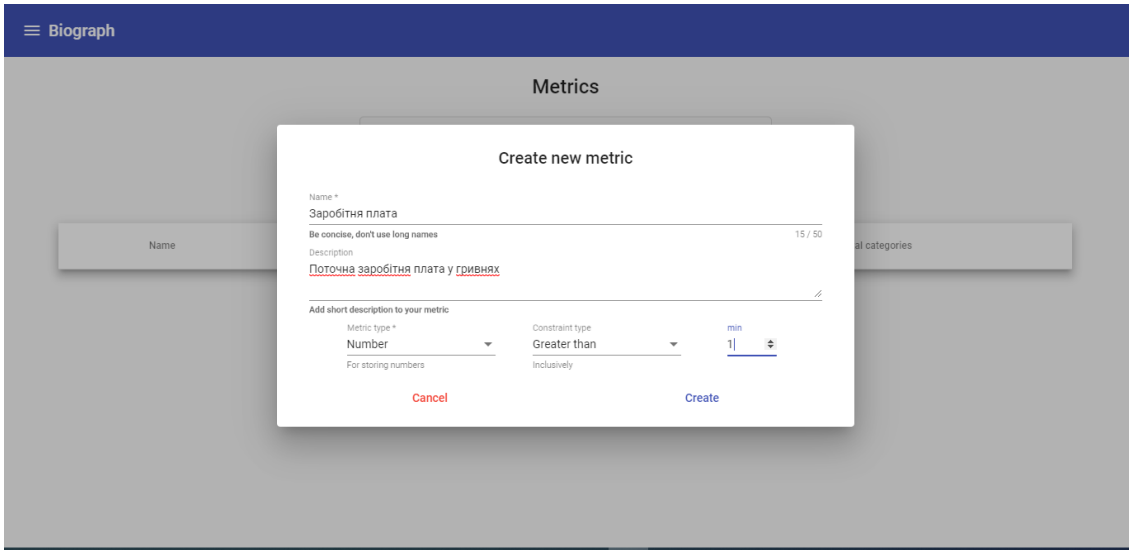


Рисунок 3.9 – Приклад заповнення форми створення метрики

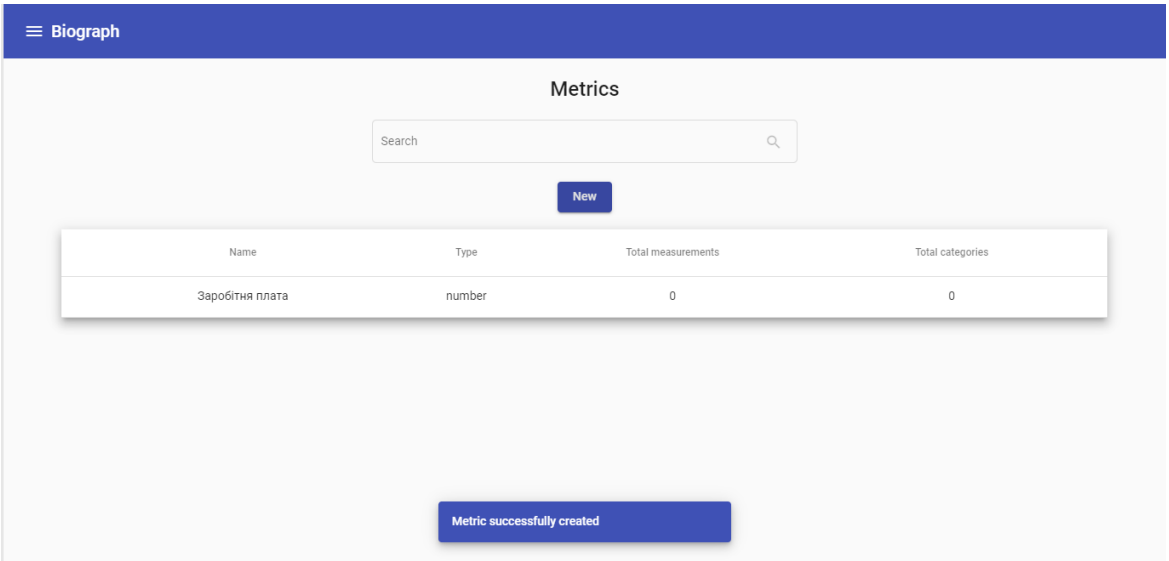


Рисунок 3.10 – Результат натискання кнопки “Create”

3.2.5 Створення категорії події

Для створення категорії події необхідно:

- відкрити бокове меню додатку;
- натиснути кнопку “Categories”;
- на сторінці “Categories”, що відкрилась, натиснути кнопку “New”;
- у діалоговому вікні, що відкрилось, заповнити всі необхідно поля форми, відповідно до інструкцій, що демонструє додаток;

- натиснути кнопку “Create”.

При створенні категорії необхідно задати перелік метрик, що відносяться до створюваної категорії.

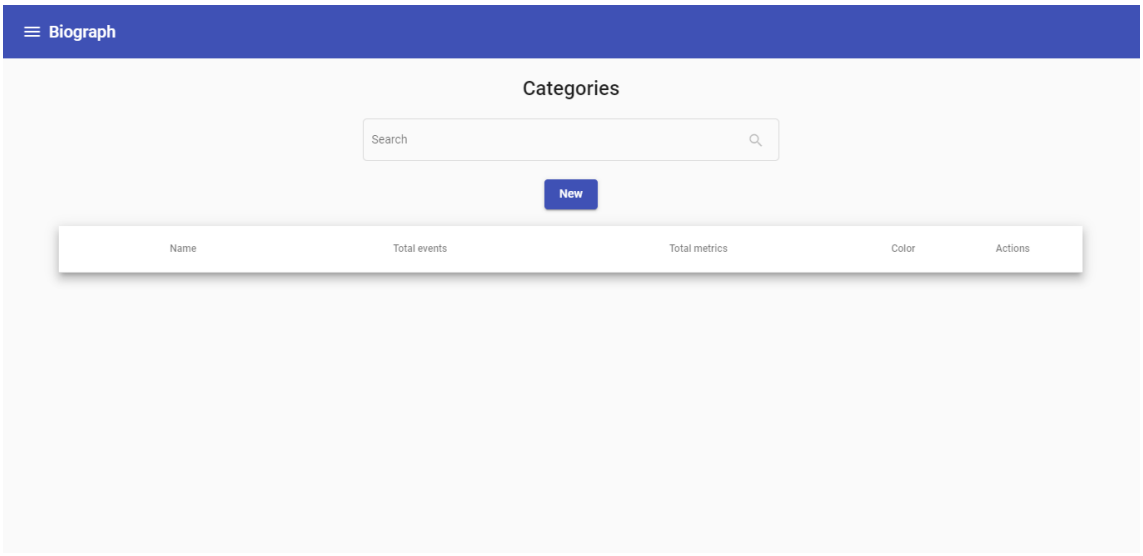


Рисунок 3.11 – Сторінка “Categories”

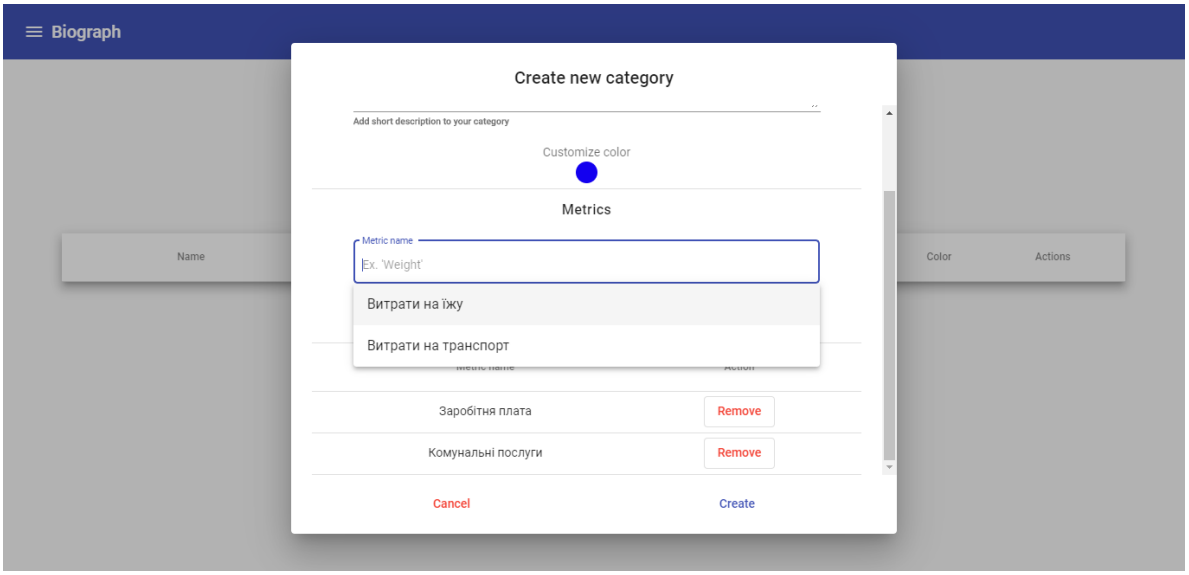


Рисунок 3.12 – Приклад заповення форми створення категорії

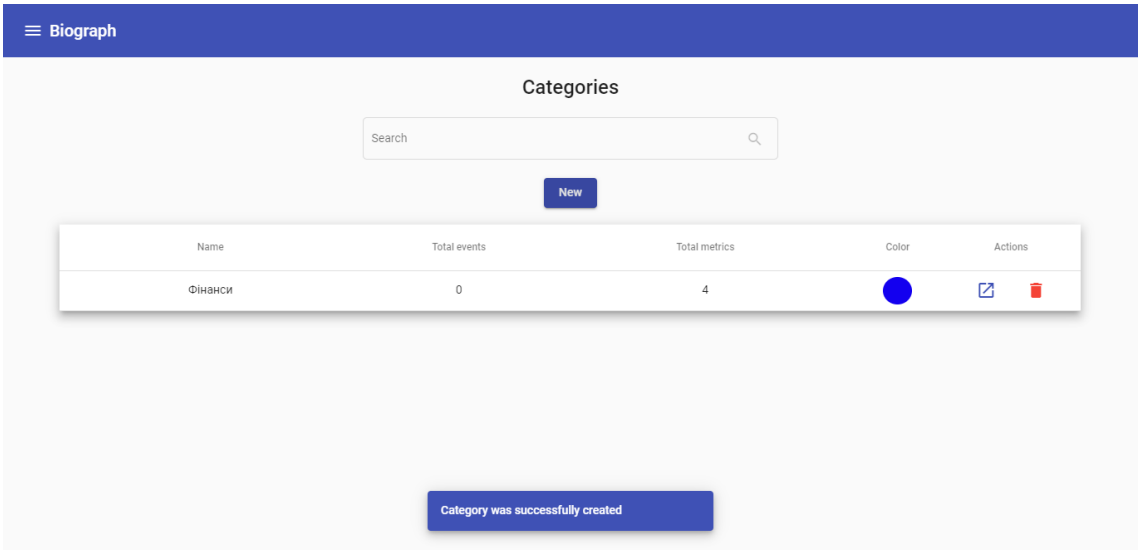


Рисунок 3.13 – Результат натискання кнопки “Create”

3.2.6 Створення події

Для створення події необхідно:

- відкрити бокове меню застосунку;
- натиснути кнопку “Events”;
- на сторінці, що відкрилась, натиснути кнопку “New”;
- заповнити необхідні поля форми, відповідно до інструкції, що демонструє додаток;
- натиснути кнопку “Create”.

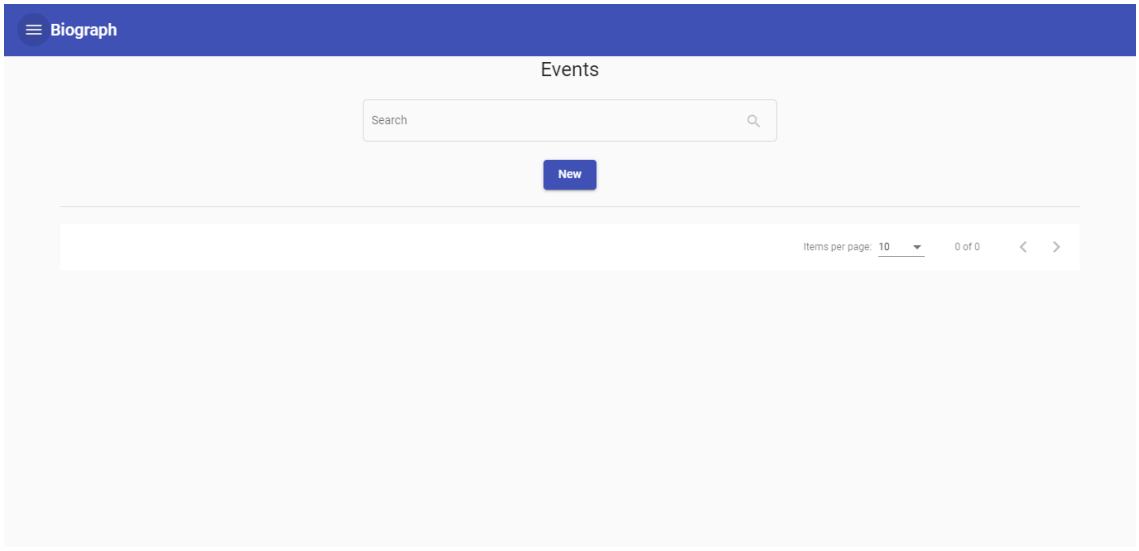


Рисунок 3.14 – Сторінка “Events”

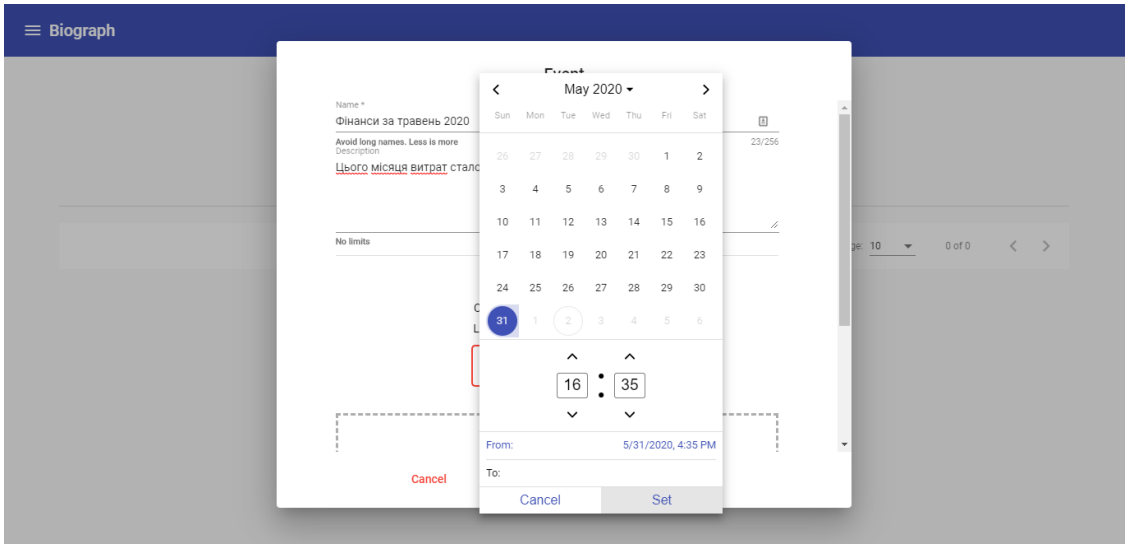


Рисунок 3.15 – Приклад введення дати події

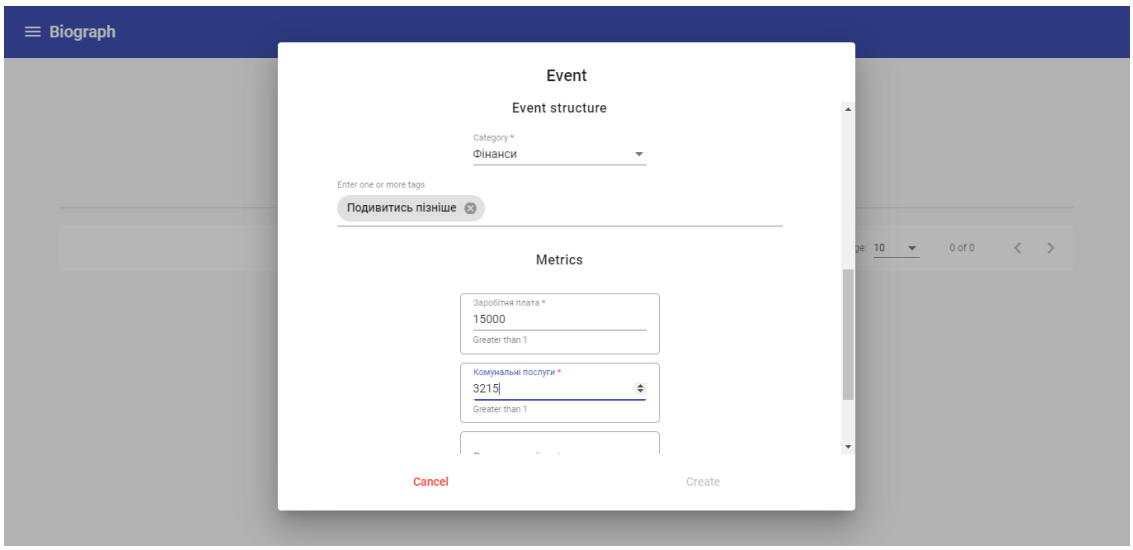


Рисунок 3.16 – Приклад визначення категорії події та значень метрик

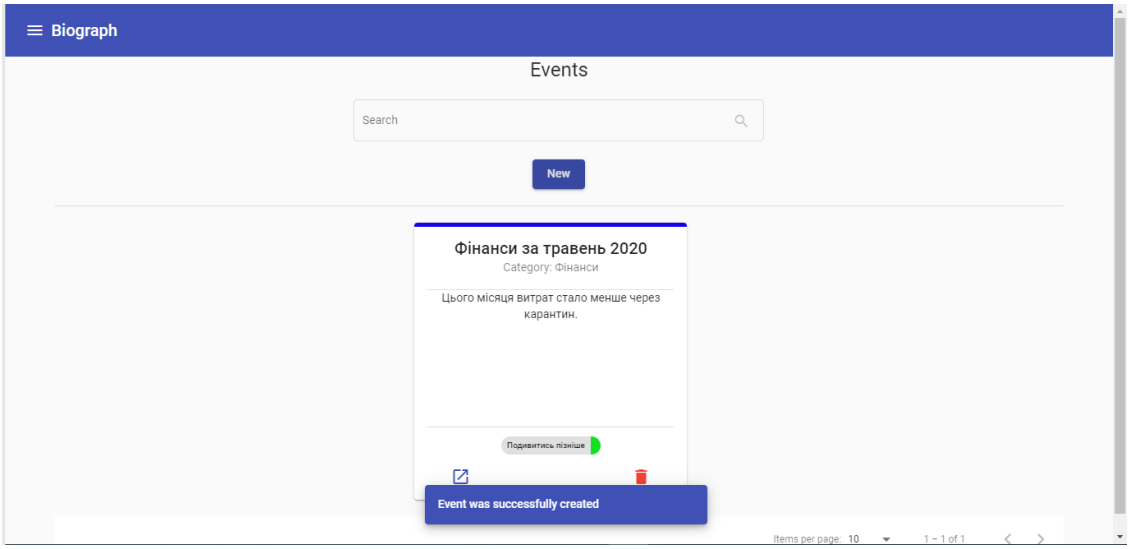


Рисунок 3.17 – Результат натискання кнопки “Create”

3.2.7 Пошук тегу

Для пошуку тегу необхідно:

- відкрити бокове меню застосунку;
- натиснути кнопку “Tags”;
- на сторінці “Tags”, що відкрилась, ввести пошуковий запит у поле пошуку.

Пошук відбувається автоматично, без необхідності натискання будь-яких інших кнопок.

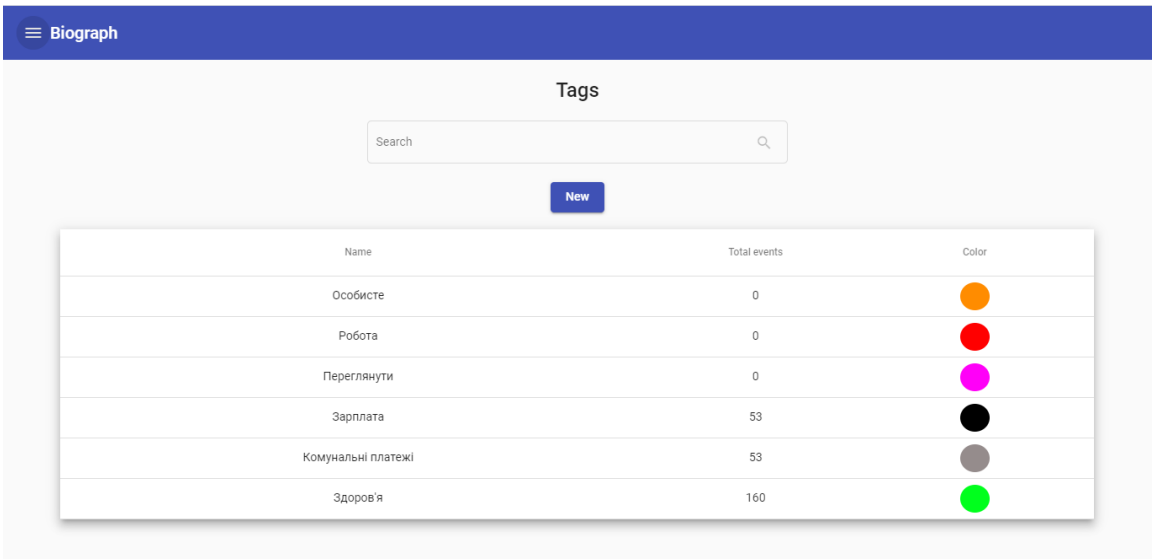


Рисунок 3.18 – Сторінка “Tags” до введення пошукового запиту

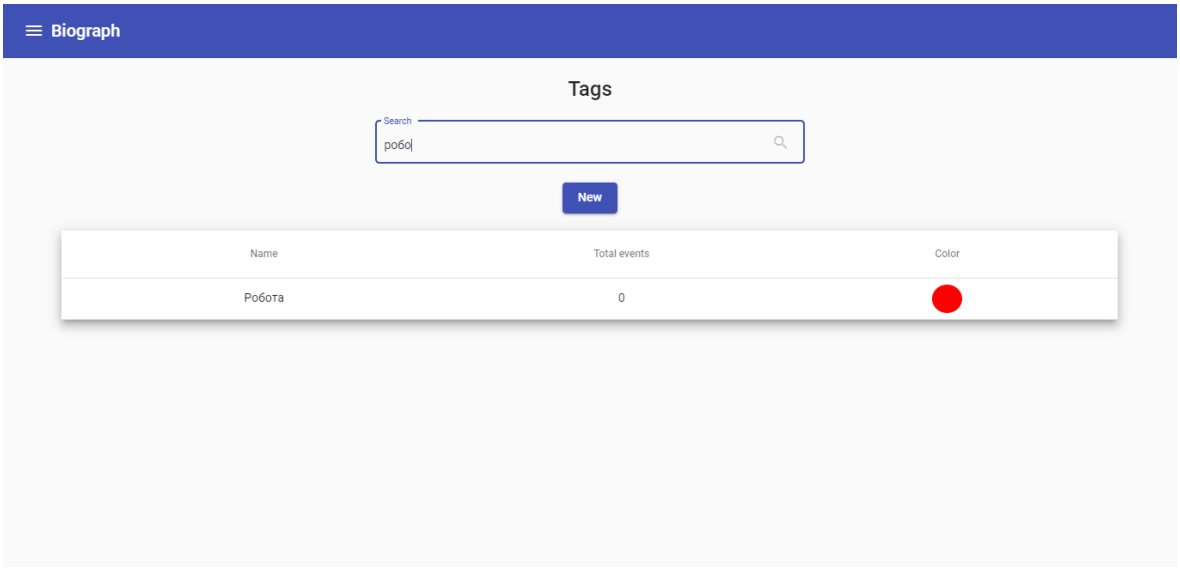


Рисунок 3.19 – Сторінка “Tags” після введення пошукового запиту

3.2.8 Пошук метрики

Для пошуку метрики необхідно:

- відкрити бокове меню застосунку;
- натиснути кнопку “Metrics”;
- на сторінці “Metrics”, що відкрилась, ввести пошуковий запит у поле пошуку.

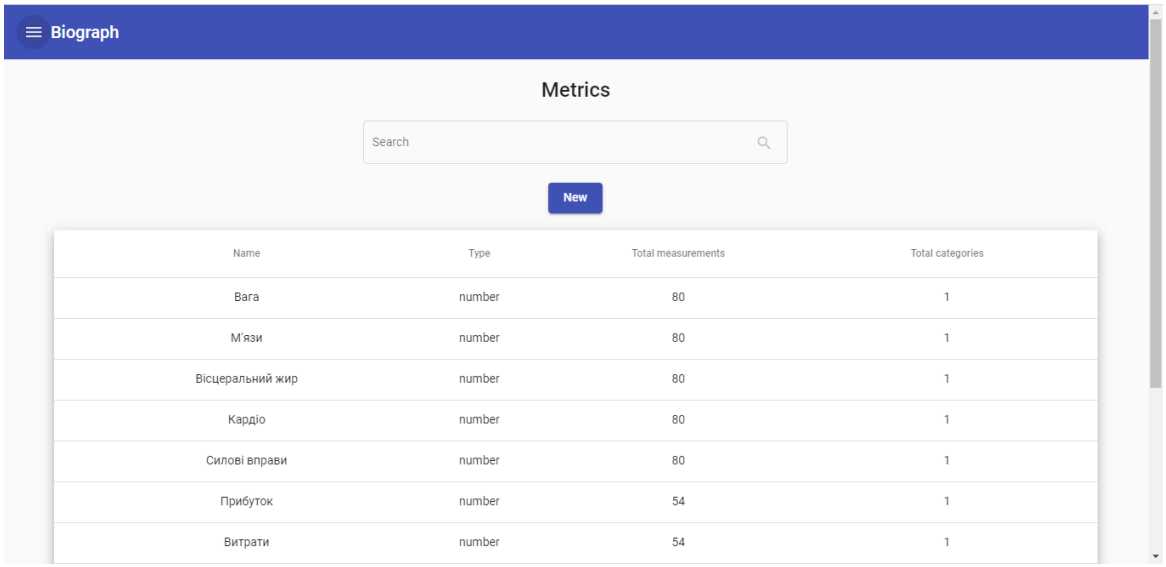


Рисунок 3.20 – Сторінка “Metrics” до введення пошукового запиту

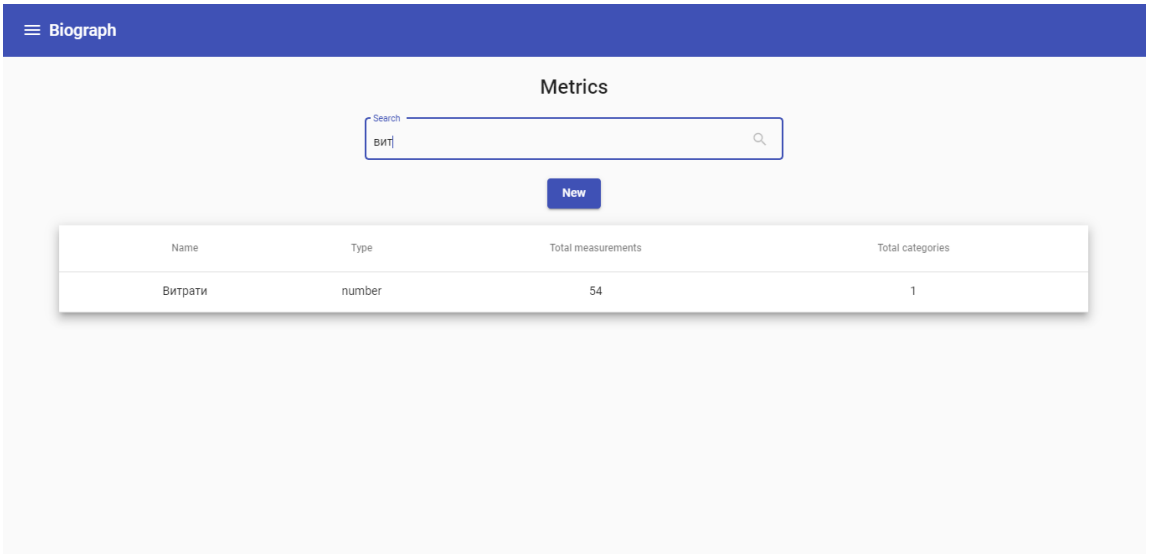


Рисунок 3.21 – Сторінка “Metrics” після введення пошукового запиту

3.2.9 Пошук категорії

Для пошуку категорії необхідно:

- відкрити бокове меню застосунку;
- натиснути кнопку “Categories”;
- на сторінці “Categories”, що відкрилась, ввести пошуковий запит у поле пошуку.

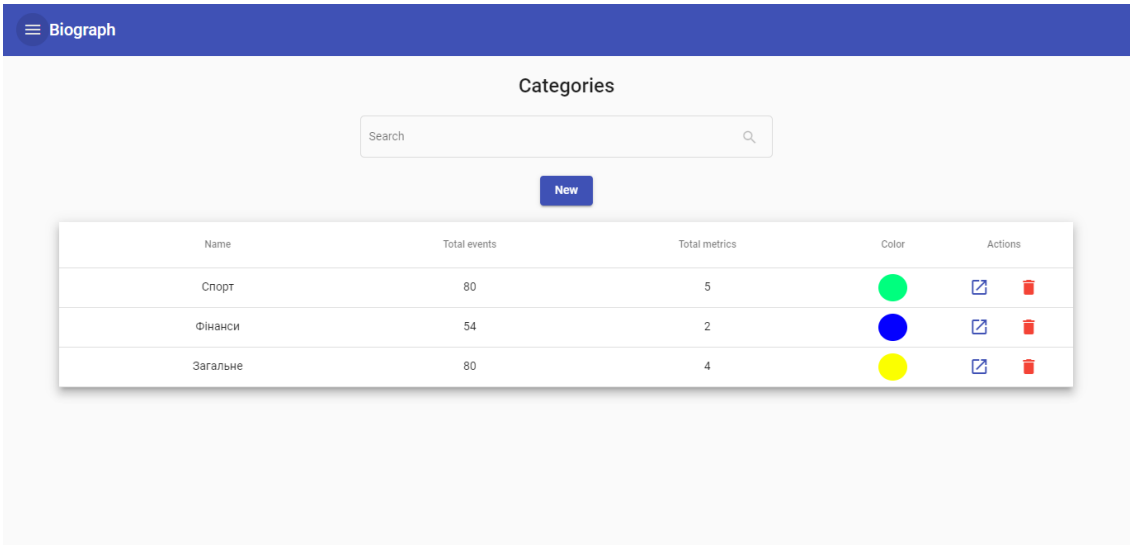


Рисунок 3.22 – Сторінка “Categories” до введення пошукового запиту

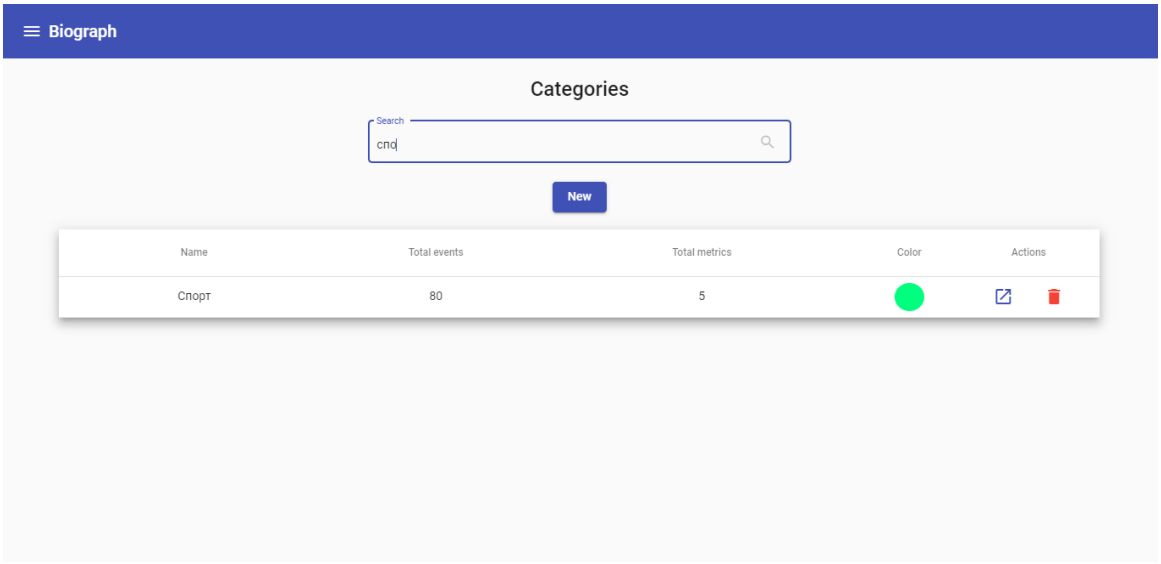


Рисунок 3.23 – Сторінка “Categories” після введення пошукового запиту

3.2.10 Пошук події

Для пошуку події необхідно:

- відкрити бокове меню застосунку;
- натиснути кнопку “Events”;
- на сторінці “Events”, що відкрилась, ввести пошуковий запит у поле пошуку.

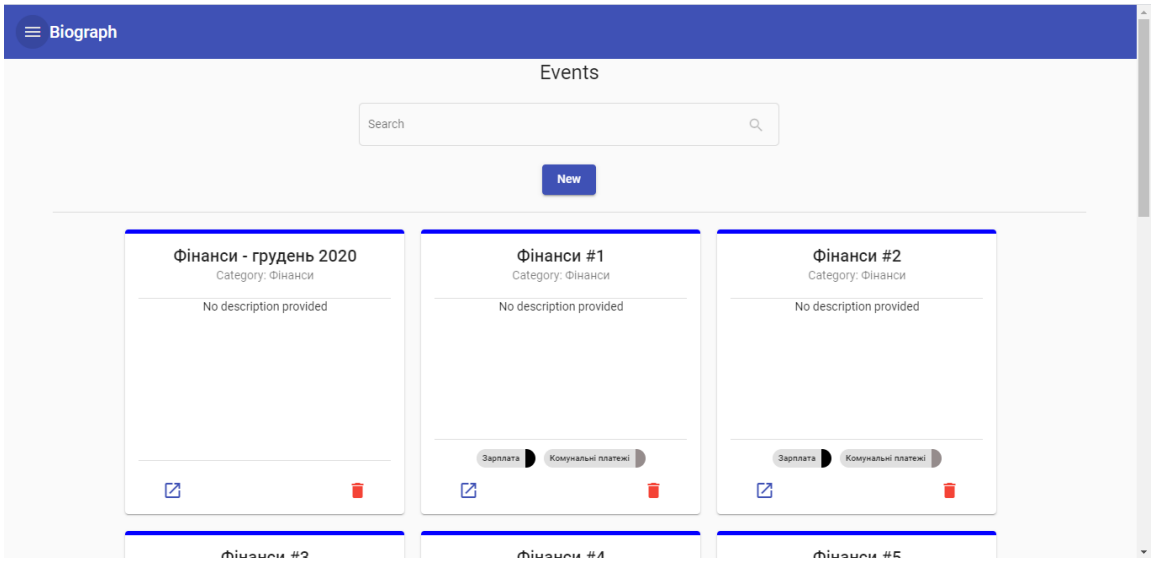


Рисунок 3.24 – Сторінка “Events” до введення пошукового запиту

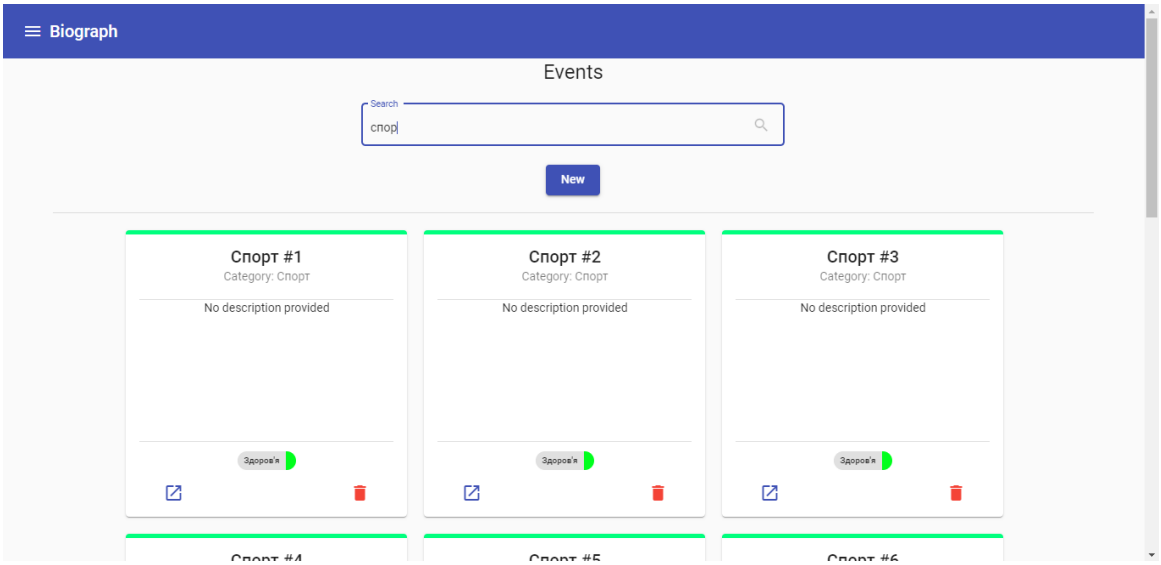


Рисунок 3.25 – Сторінка “Events” після введення пошукового запиту

3.2.11 Перегляд кругової діаграми за кожною категорією

Для перегляду кругової діаграми за кожною категоріює необхідно:

- відкрити бокове меню застосунку;
- натиснути кнопку “Statistics”;
- натиснути у переліку вкладок застосунку кнопку “By categories”;
- за допомогою слайдерів обрати необхідний проміжок часу.

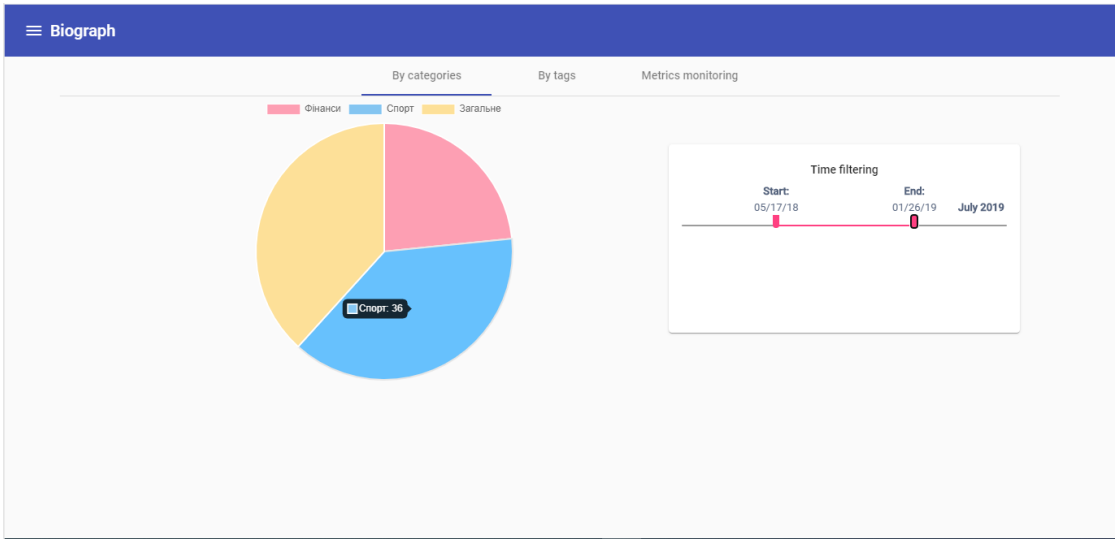


Рисунок 3.26 – Кругова діаграма кількості подій за кожною категорією

3.2.12 Перегляд кругової діаграми за кожним тегом

Для перегляду кругової діаграми подій за кожним тегом необхідно:

- відкрити бокове меню застосунку;
- натиснути кнопку “Statistics”;
- натиснути у переліку вкладок застосунку кнопку “By tags”;
- за допомогою слайдерів обрати необхідний проміжок часу.

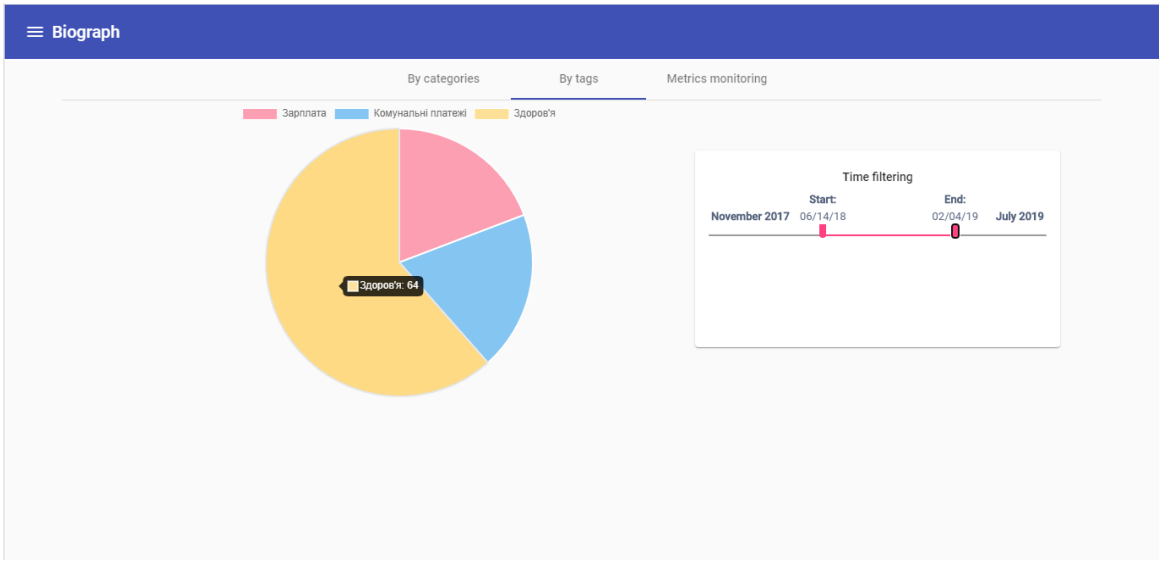


Рисунок 3.27 – Кругова діаграма кількості подій за кожним тегом

3.2.13 Перегляд графіку значень метрик у часі

Для перегляду значень метрик у часі необхідно:

- відкрити бокове меню застосунку;
- натиснути кнопку “Statistics”;
- натиснути кнопку “Metrics monitoring”;
- для кожної метрики, значення якої потрібно відобразити на графіку:
  а) обрати метрику з випадаючого списку;
  б) увімкнути перемикач “Display metric”;
  в) з випадаючого списку обрати агрегуючу функцію;
  г) за необхідності, увімкнути перемикач “Normalize metric”;
  д) з випадаючого списку обрати нормалізуючу функцію.
- натиснути кнопку “Apply”.

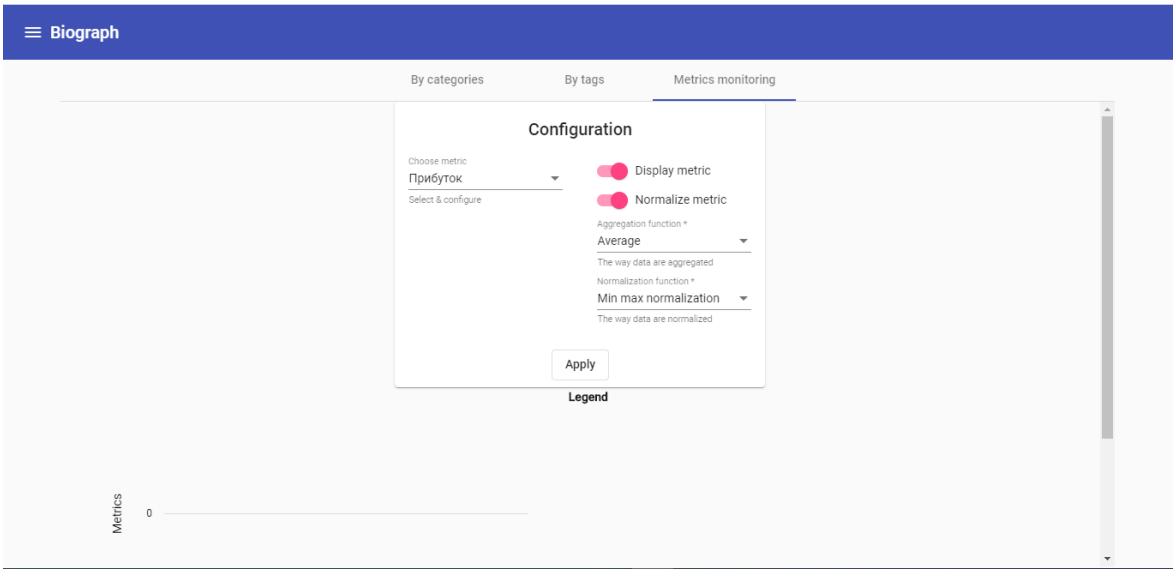


Рисунок 3.28 – Приклад налаштування відображення значень метрики

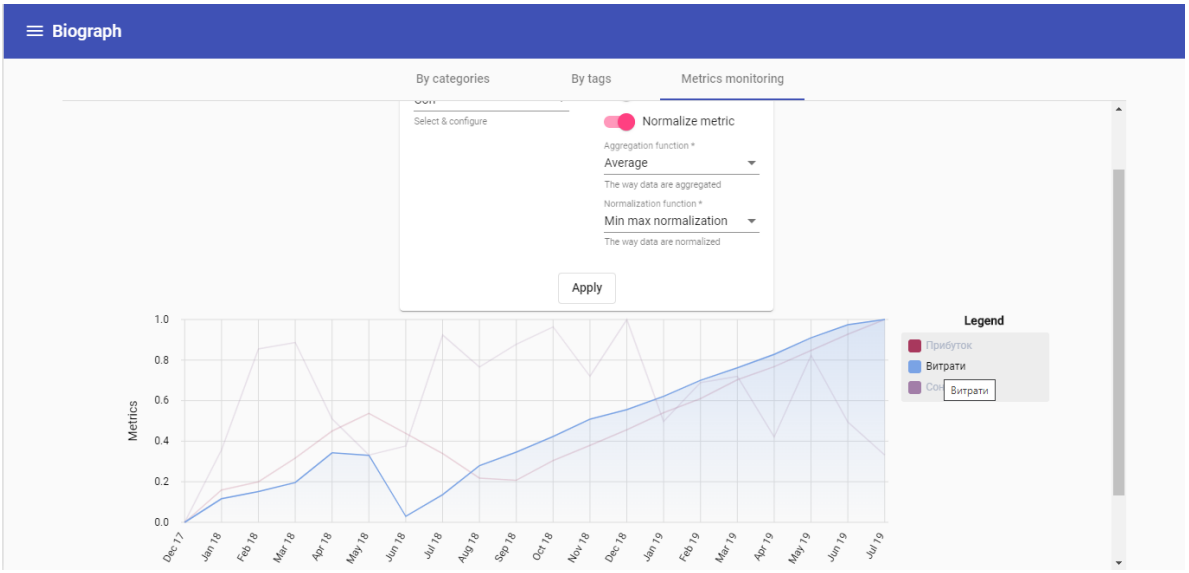


Рисунок 3.29 – Демонстрація графіку зміни значень метрик у часі

3.3 Завершення роботи

Для завершення роботи з програмним забезпеченням, оператор повинен відкрити бокове меню та натиснути кнопку “Sign out”. Після цього необхідно закрити вкладку браузера, у якій було відкрито програмне забезпечення. Детальна інформація, про способи закриття вкладки браузера наведена у керівництві користувача браузера.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

ВЕБ-ЗАСТОСУВАННЯ МОНІТОРИНГУ ЖИТТЄВИХ ПОДІЙ

Графічний матеріал

КПІ.ІП-6326.045420.07.99.СС

“ПОГОДЖЕНО”

Керівник проекту:

_____О.В. Ковтунець

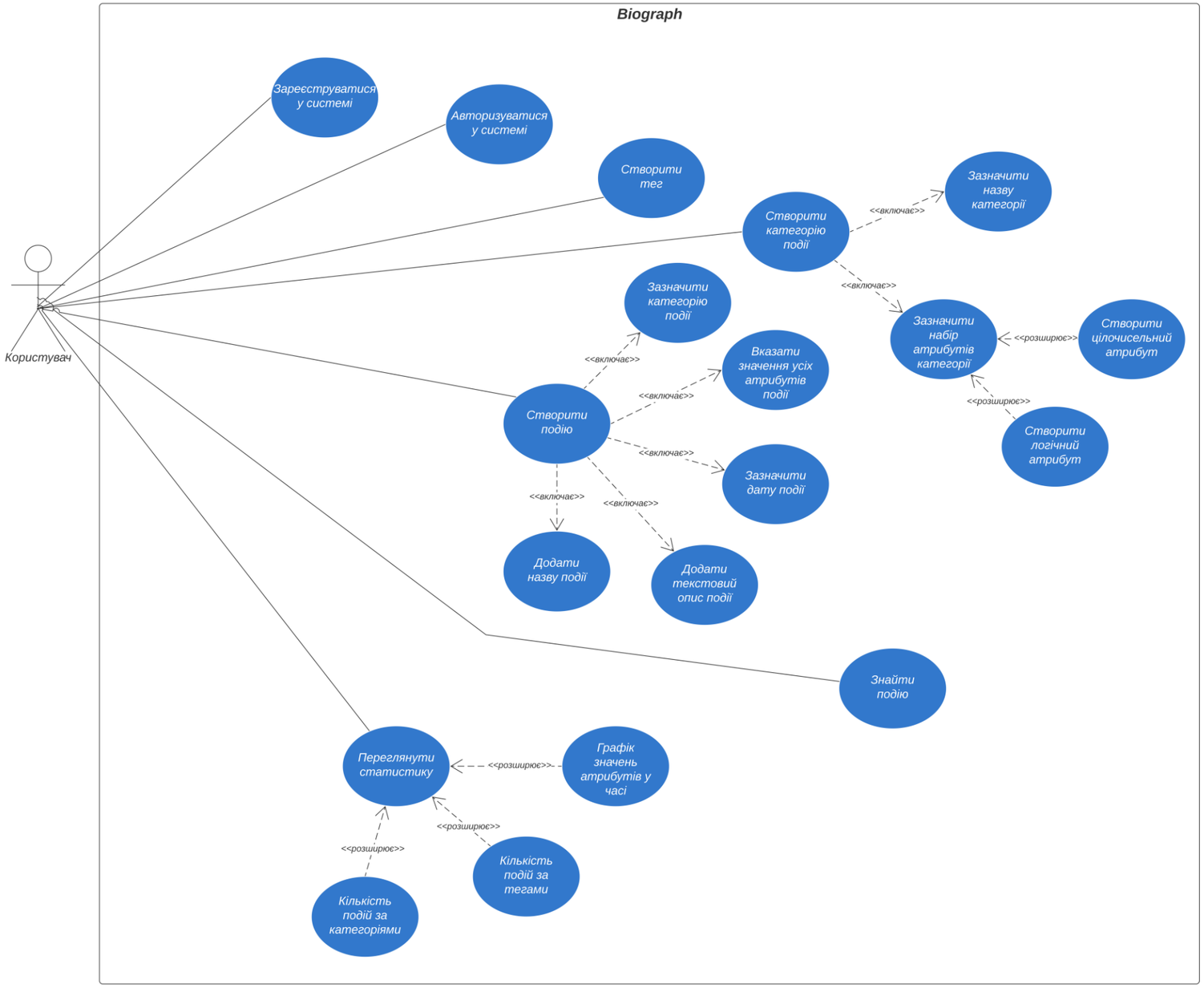
Нормоконтроль:

_____К.І. Ліщук

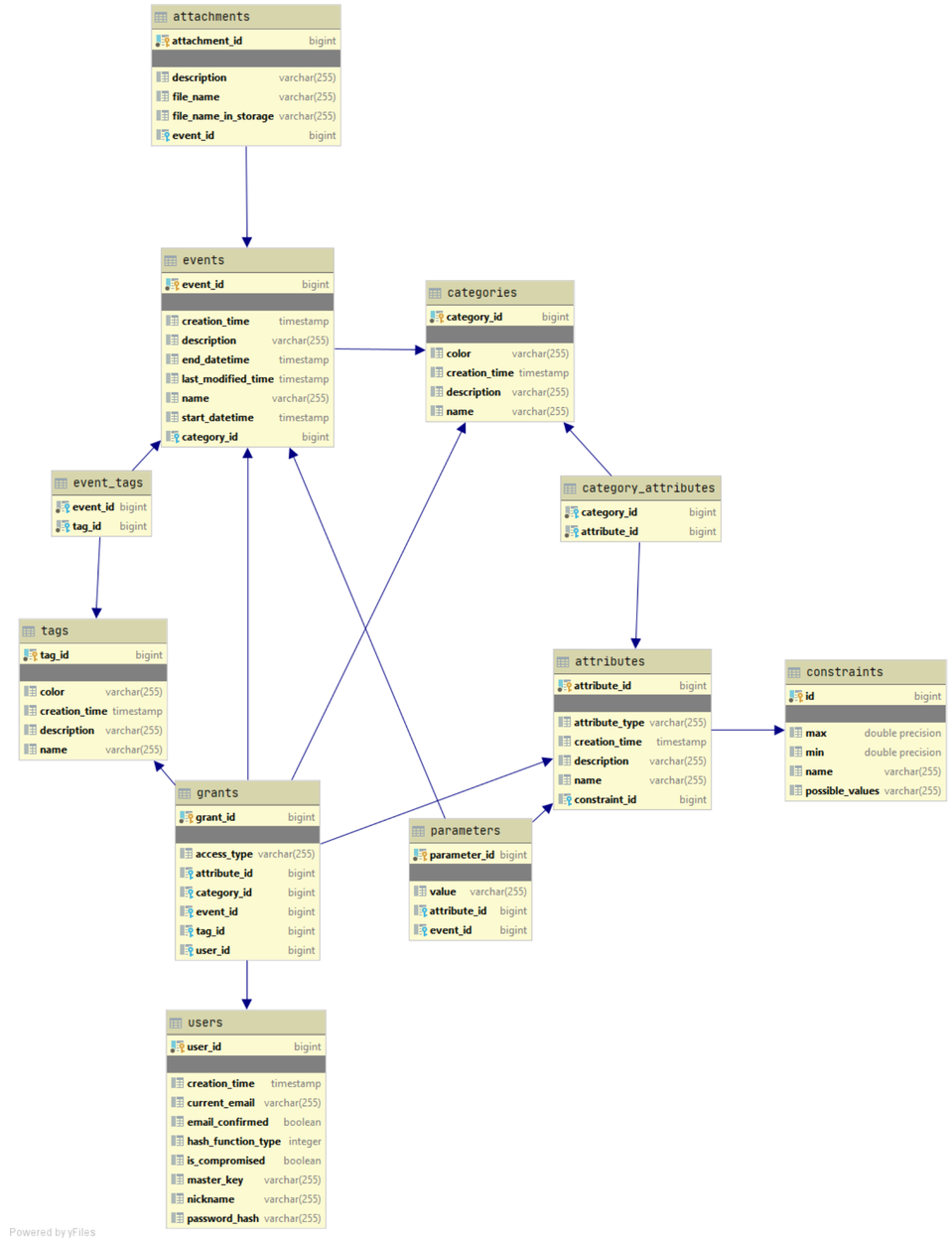
Виконавець:

_____А.О. Семченко

Київ – 2020 року

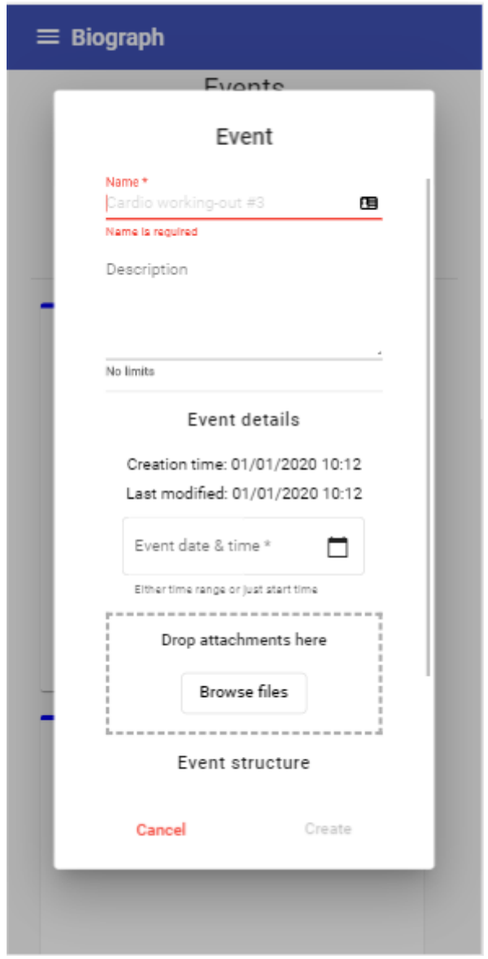


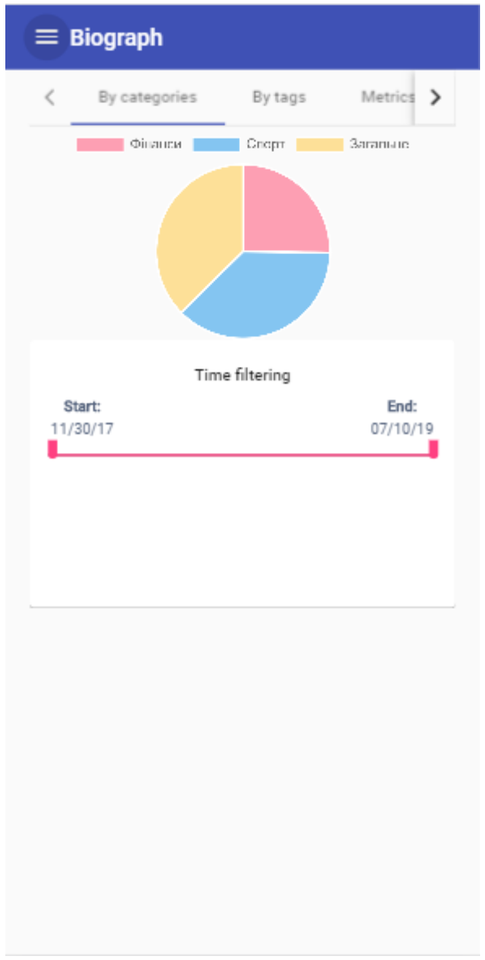
					КПІ.ІП-6326.045420.07.99.СС		
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна варіантів використання	Літера	Маса / Масштаб
Розробив		Семченко А.О.					
Перевірив		Ковтунець О.В.				Аркуш	Аркушів
Т. кон.							
Н. кон.		Ліщук К.І.			Веб-застосування моніторингу життєвих подій	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-63	
Затвердив		Ковтунець О.В.					

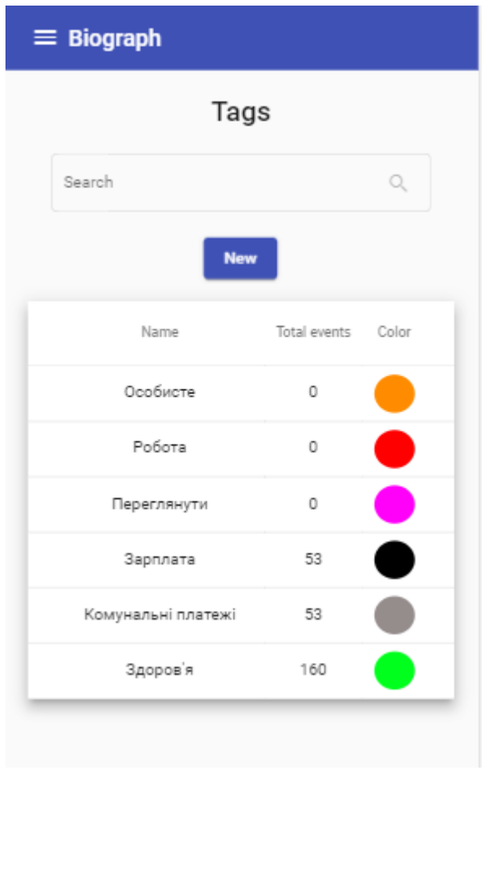


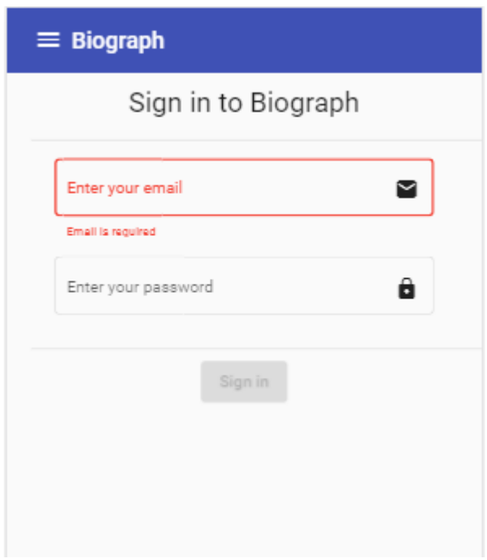
Інв. № підл.	Підп. дата	Інв. № взам. підл.	Інв. № дубл.	Підп. дата

Зм.	Арк.	№ докум.	Підп.	Дата	КПІ.ІП-6326.045420.08.99.СБД		
Розроб.		Семченко А.О.			Схема бази даних	Лит.	Арк.	Аркушів
Перев.		Ковтунець О.В.						1
Т. Кон.						Аркуш		Аркушів
Н. Кон.		Ліщук К.І.			Веб-застосування моніторингу життєвих подій	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-63		
Затв.		Ковтунець О.В.						

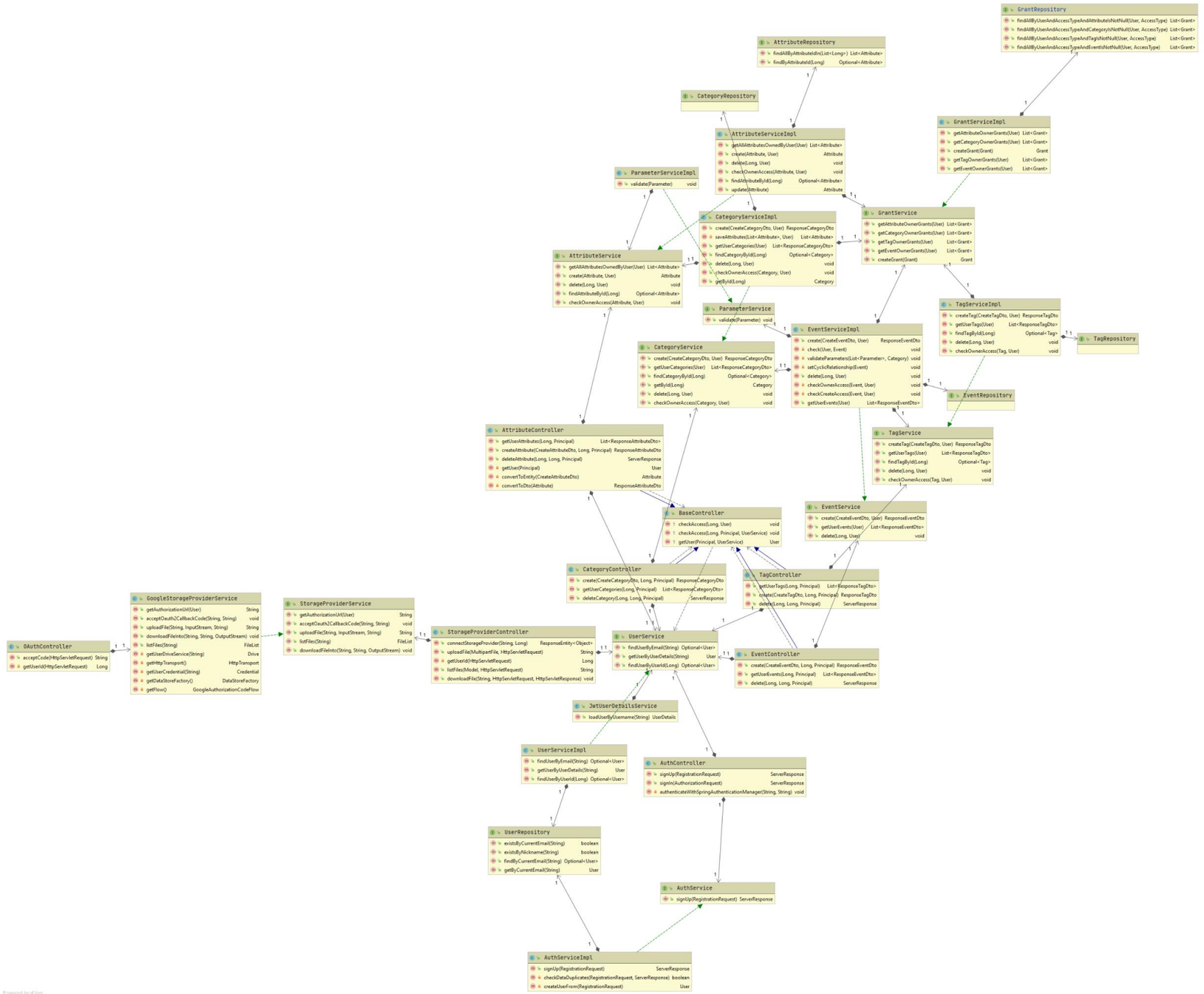








					КПІ.ІП-6326.045420.09.99.КЕ	
						Літера / Маса / Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	Креслення вигляду екранних форм	
Розробив		Семченко А.О.				Аркуш / Аркушів
Перевірив		Ковтунець О.В.				
Т. кон.						
Н. кон.		Ліщук К.І.			Веб-застосування моніторингу життєвих подій	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-63
Затвердив		Ковтунець О.В.				



					КПІ.ІП-6326.045420.10.99.СС	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна класів програмного забезпечення	Літера / Маса / Масштаб
Розробив		Семченко А.О.				
Перевірив		Ковтунець О.В.				Аркуш / Аркушів
Т. кон.						
Н. кон.		Ліщук К.І.			Веб-застосування моніторингу життєвих подій	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-63
Затвердив		Ковтунець О.В.				

КПІ.ІП-6326.045420.10.99.СС